Sculpture virtuelle

Eric Ferley

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

T H È S E

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : "Imagerie, Vision et Robotique"

préparée au laboratoire *i*MAGIS-GRAVIR
dans le cadre de **l'École Doctorale "Mathématiques, Sciences et Technologie de l'information"**

présentée et soutenue publiquement

par

Eric FERLEY

le 18 septembre 2002.

Titre : Sculpture Virtuelle

Directrice de thèse : Marie-Paule CANI
Co-directeur : Jean-Dominique GASCUEL

JURY

M.	James L.	CROWLEY	, Président
M.	Sabine	COQUILLART	, Rapporteur
M.	Christophe	SCHLICK	, Rapporteur
M.	Marie-Paule	CANI	, Directrice de thèse
M.	Jean-Dominique	GASCUEL	, Co-encadrant
M.	Andras	KEMENY	, Examinateur

Remerciements

Je tiens à remercier Claude Puech pour sa gentillesse et cette petite bulle de bonne humeur et de dynamisme au sein de laquelle j'ai passé quelques années : *iMAGIS* a vraiment été un épisode marquant et agréable, à la source de rencontres sensationnelles !

Je remercie également Sabine Coquillart et Christophe Schlick pour avoir accepté d'être les rapporteurs de ma thèse, et Jim Crowley pour avoir accepté de présider mon jury.

Enfin, je remercie Andras Kemeny pour avoir accepté de faire partie de mon jury, et pour ses encouragements à rédiger et soutenir. J'associe à ces remerciements Stéphane, qui a su, avec Andras me laisser un peu de liberté et de temps pour y parvenir, pendant cette période où je cherchais à afficher des voitures.

Je tiens tout particulièrement à remercier Marie-Paule et Jean-Dominique pour avoir su me pousser à sortir de mes ornières et supporté mes errements. Merci de votre soutien et de vos conseils à tous deux.

Je ne citerais pas les noms de toutes les autres personnes formidables rencontrées durant ce long cheminement, et sans qui ce document n'aurait sans doute pas vu le jour . . . vous vous reconnaissez, et vous savez déjà.

J'adresse ici aussi un grand merci à ma famille pour son soutien et sa chaleur malgré les turbulences et les doutes durant toutes ces années . . .

SOMMAIRE

Introduction

Contexte

Cette thèse s'inscrit dans le cadre d'une collaboration avec Renault. Sa participation conjointe avec le CNRS a permis sa réalisation. Le thème général des travaux de la thèse concerne la modélisation de formes tridimensionnelles.

Motivations

On s'intéresse ici à la modélisation de formes par un utilisateur habitué à une interaction tridimensionnelle *directe* avec un matériau réel. Une interaction bi-dimensionnelle *classique* à travers un écran et un dispositif d'entrée 2d tel qu'une souris ou un stylo et des courbes de contours, nécessite une manipulation *indirecte* via des points de contrôles ou des zones de sélections plus ou moins complexes ainsi que plusieurs vues simultanées pour appréhender la surface modélisée. De telles *indirections* contribuent à notre sens à distraire de la forme en construction.
En réponse à ces remarques, des approches comme SKETCH ou Teddy, sur lesquelles nous reviendrons, permettent en exploitant la gestuelle lors du dessin de se rapprocher d'une interaction de type crayon papier pour obtenir rapidement des formes simples et approximatives. Toutefois, elles sont encore relativement limitées et ne permettent pas d'affiner les formes obtenues une fois que l'on en est satisfait.
Nous pensons qu'une interaction tridimensionnelle directe, comme le modelage d'argile ou de pâte à modeler, permet un meilleur contrôle, c'est-à-dire à la fois d'obtenir des formes grossières et de les affiner progressivement pour, dans la continuité, aboutir éventuellemnt à une forme assez complexe et détaillée par itérations. Ce type d'interaction tridimensionnelle peut apparaître *moins intuitive* qu'une interaction bidimensionnelle : tout le monde sait faire un croquis, mais fabriquer des formes $3D$ par la sculpture ou la poterie / pâte à modeler requiert un apprentissage. C'est cette aptitude à la manipulation et à l'évaluation que nous qualifions d'entrainement à l'interaction tridimensionnelle directe, et supposons acquise par l'utilisateur. Notre démarche consiste à se rapprocher le *plus possible* de cette situation d'interaction directe avec une surface.

Difficultés

Plusieurs problèmes se posent pour y parvenir, tant du côté de la restitution/perception (aspects visuels stéréoscopiques et haptiques) que du côté de la description géométrique sous-jacente. En effet, cette représentation pour ne pas distraire ou gêner doit pouvoir être ignorée lors de l'interaction et permettre des changements tels que la déconnexion / reconnexion sans complications.

Nous verrons dans la première partie de ce document que la plupart des représentations nécessitent des actions sur leurs paramètres internes et / ou font appel à des structures de contrôle qui peuvent devenir très complexes et difficiles à manipuler.

Approche et buts

Dans ce contexte, une approche par surfaces implicites dont les avantages comprennent justement la gestion aisée des changements de topologie nous a semblé d'emblée prometteuse. Toutefois, les limitations inhérentes à cette formulation semblaient au départ de cette thèse contredire cette intuition. En effet, l'affichage et la mise à jour de ces surfaces est gourmande en ressources machine. Une formulation classiquement utilisée dans le domaine de la synthèse d'images repose sur une décomposition en *squelettes* plus ou moins complexes. Une telle formulation conduirait dans notre cas à un stockage itératif lors d'une session de modélisation, et se révèlerait rapidement prohibitive tant en termes de stockage qu'en temps de calculs.

Contributions

C'est pour cette raison que nous nous sommes orientés vers un stockage discret de ces surfaces implicites, cette étape étant de toute façon nécessaire à l'extraction et l'affichage des surfaces. Qui dit discrétisaton, dit forcément échantillonnage et filtrage. Nous avons considéré ces limitations non bloquantes dans notre première maquette mono-résolution. Notre seconde maquette multi-résolution permet d'adapter automatiquement la résolution d'échantillonnage pour contourner ces problèmes et autoriser l'ajout de détails aussi fins que souhaité. La multirésolution permet de plus de gérer la complexité de l'affichage ainsi que la mise à jour en tâche de fond pour maintenir l'interactivité et assurer un retour rapide des modifications en cours d'édition.

Organisation du document

Dans le présent document, nous passons en revue dans une première partie les différentes *armes* dont nous disposons pour affronter les problèmes posés. Nous nous consacrons d'abord aux différentes représentations polygonales, paramétriques, etc ... permettant de générer des formes $3D$. Cela permet de mieux situer et justifier notre choix des surfaces implicites. Nous abordons ensuite les différents types d'interaction ainsi que les dispositifs d'entrée et de restitution qui leur sont liés.

Dans la seconde partie, nous présentons notre première contribution permettant un stockage efficace et une mise à jour interactive dans le cas d'un échantillonnage mono résolution. C'est sur cette maquette que nous avons également exploré les dispositifs de restitution stéréo et de retour d'effort.

Enfin dans la troisième partie, nous présentons notre maquette adaptant la résolution pour échantillonner *correctement* la surface et maintenir l'interactivité lors de l'affichage et de la mise à jour.

Finalement, nous concluons en reprenant nos contributions et en détaillant les pistes que nous n'avons pu explorer et qui nous semblent prometteuses dans d'éventuelles poursuites de ces travaux.

CHAPITRE 1

Travaux antérieurs

L'objet de cette partie est de préciser la notion de *sculpture virtuelle* qui apparaît assez tôt dans la littérature infographique [Par77] et est employée pour désigner une vaste gamme de travaux plus ou moins éloignés de ce que nous entendons par ce terme. Cette partie nous permet également de situer nos travaux parmis les différentes approches existantes, et de justifier quelques uns de nos choix initiaux. Nous essaierons de retracer brièvement l'évolution de cette métaphore de sculpture en tentant de séparer les deux aspects majeurs à notre sens : modélisation et interaction.

Dans la première partie, nous allons nous concentrer sur l'aspect **modélisation** à travers quelques articles marquants, car l'exhaustivité en ce domaine constituerait un livre à part entière. Par modélisation, nous entendons à la fois la description du modèle et de ses possibilités d'édition, qui sont souvent intimement liées, comme on le verra. On s'intéressera donc dans cette partie également aux **manipulations**, **éditions** ou **modifications** qu'autorise un modèle. Notre cheminement sera évidemment *orienté* vers le modèle que nous avons choisi ; nous passerons plus ou moins vite sur des travaux néanmoins intéressants, ou apparus pendant ou après nos travaux.

La seconde partie, consacrée à l'**interaction**, est plutôt axée sur les *technologies* permettant d'interagir *physiquement* avec le modèle manipulé. On s'intéressera ici aux dispositifs et procédés permettant aussi bien de **percevoir** que d'**agir** sur le modèle.

1 Modélisation

Nos travaux concernent donc pour une part la production de formes tridimensionnelles, à partir de formes existantes ou non. Précisons dès maintenant que le produit résultat d'une session de *sculpture* est ainsi une surface ou *peau*. A priori cette *peau* est statique et sans élément structurant (un squelette de déformation, ou une hiérarchie articulée par exemple) comme peuvent en produire certains logiciels de modélisation en vue d'animer cette peau. Nous faisons ce choix pour séparer les problèmes :

on s'intéresse ici à la modélisation d'une forme, et pas à une modélisation dans un cadre/but donné[1].

Nous ne chercherons pas à élaborer une nouvelle classification des différentes représentations de surfaces. On pourra pour cela se reporter au livre [FvDFH90] ou à l'un des nombreux cours disponibles sur le web. Pour fixer les idées, et aussi pour situer l'éventail que nous allons *visiter* dans cette partie, la Figure 1.1 présente une classification possible des entités 3*D* manipulables en infographie.

- Données brutes
  - nuage de points
  - image de profondeur
  - soupe de polygones
- Surfaces
  - maillage
  - subdivision
  - paramétrique
  - implicite
- Solides
  - voxels
  - arbre *BSP*
- Structure de haut niveau
  - *Constructive Solid Geometry*
  - modèles générateurs (fractales ?)
  - squelette
  - extrusions
  - graphe de scène

FIG. 1.1: Une classification possible des entités 3*D* utilisées en infographie (traduite à partir du cours C0S 598B de T. Funkhouser, université de Princeton, été 2000).

Notre objectif à priori est de trouver une solution pour (1) construire et éditer interactivement une surface, en (2) cachant la représentation sous-jacente. Cela dans le but de reproduire une *métaphore de sculpture*, où l'utilisateur interagit à travers un outil, il faudra donc sans doute (3) que la représentation permette également de détecter efficacement des collisions, déduire des forces de réactions ...

Il est clair qu'aucune représentation ne saurait satisfaire tous les besoins. Notre besoin particulier ici est l'édition, si possible *directe*, c'est-à-dire le plus loin possible de la représentation, justement. Plutôt que de passer en revue chaque représentation de surface, nous proposons plutôt de cheminer entre différentes contributions qui nous semblent marquantes pour illustrer certains principes d'édition et souligner leurs intérêts ou limitations.

A part de rares et récentes exceptions comme le rendu volumique, le ray-tracing / splatting interactif ou le rendu par points, les dispositifs d'affichage 3D (cartes graphiques) utilisent comme primitive de base le triangle[2]. Ainsi, pour pouvoir afficher une surface, quelle que soit sa description (paramétrique, implicite, etc...), il faut à un moment donné forcément la **convertir** en un modèle polygonal (opération de triangulation). Ce modèle est donc le passage obligé de toutes les représentations imaginables. D'où l'idée de travailler directement sur des surfaces polygonales, intersection commune de tous les modèles.

1.1 Edition directe de Surfaces polygonales

L'approche la plus directe pour éditer des surfaces polygonales est de le faire sommet par sommet. Cette approche montre vite ses limites face au nombre de sommets On peut imaginer aussi de sélectionner des groupes de sommets pour réduire le nombre de manipulations. Cela ne résoud pas forcément la complexité et ne procure pas de plus un contrôle très souple. Une autre solution, comme

[1] Cet argument est de plus appuyé par d'autres travaux de recherche visant à animer des *peaux* statiques, motivés notament par les progrès sur l'acquisition directe d'objets réels. Ce n'est par conséquent pas un problème bloquant qui interdirait toute utilisation à postériori des formes produites pour par exemple les animer.

[2] Pour être exact, OpenGL autorise plus généralement les polygones, mais ces derniers sont en fait découpés en triangles arbitrairement au moment du rendu.

par exemple dans le système que nous décrivons ci-après consiste à utiliser une sélection *implicite* par un voisinage d'un sommet sélectionné.

Interactive and Controlled Synthesis of 3D Irregular Shapes

L. Moccozet et P. Kalra utilisent dans [MK93] le concept des fonctions d'influence introduit par R. Parent dans [Par77]. Lors du déplacement d'un sommet, ceux de ses voisins contenus dans la région d'influence suivent le même déplacement pondéré par cette fonction d'influence. Au niveau des opérations qu'ils utilisent les auteurs distinguent deux catégories :

- opérations locales : un sommet, appelé *apex*, est sélectionné. Il peut ensuite être *tiré* ou *poussé*. Les sommets avoisinants suivent le même déplacement, pondéré par une *fonction d'influence* (voir Figure 1.2) dépendant de leur distance à l'apex.
- opérations globales : les déformations de [Bar84], ou les FFD (voir section 1.3) peuvent être appliquées à une région sélectionnée de l'objet, ou à l'objet entier.

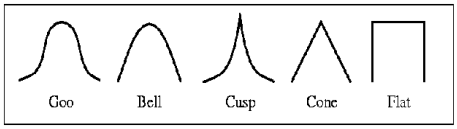


FIG. 1.2: Fonction d'influence *decay functions* : le déplacement des sommets voisins de l'*apex* décroît selon leur éloignement, et selon la forme de la fonction d'influence choisie (figure extraite de [BL95]).

Les auteurs précisent également l'importance du contrôle de la résolution dans le processus, c'est-à-dire du nombre de polygones. Ils s'intéressent surtout à l'aspect simplification pour assurer l'interactivité, ou réduire la complexité d'un modèle polygonal.

On se trouve toutefois ici dans une situation où l'on travaille plus au niveau triangle / sommet qu'au niveau de la forme elle-même. Ce type d'approche semble plus adapté à la mise au point fine (ou petites modifications) d'une forme existante qu'à la création (ou modifications importantes). Une idée reprise dans les deux approches suivantes consiste à abstraire l'interaction avec le maillage par le biais d'un *outil virtuel*.

SCULPT : an Interactive Solid Modeling Tool

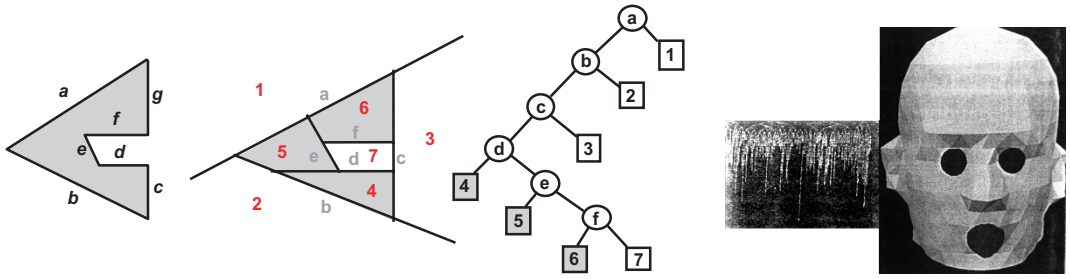
B. Naylor propose dans [Nay90] d'éditer un modèle polygonal à l'aide d'un outil :

- dont la représentation est choisie parmi huit formes simples (cube, cône, sphère, cylindre), de taille et de couleur contrôlable, et pouvant posséder comme attribut une texture.
- qui agit en réalisant une opération CSG comme l'union, l'intersection ou la différence sur le modèle.

Tous les objets de la scène (l'outil et le modèle) sont stockés sous forme d'arbres de partitionnement binaires (*BSP-Tree*) pour accélérer les calculs de localisation/collision et réaliser les opérations CSG. Les plans des facettes du modèle sont utilisés pour partitionner l'espace lors de la construction de l'arbre *BSP* (voir Figure 1.3.a).

Cette formulation impose l'emploi d'outils convexes ; les auteurs précisent qu'il est possible d'utiliser des outils non convexes, mais que la complexité de la mise en œuvre devant les besoins à l'usage rend cette approche inutile.

L'outil peut être positionné par l'utilisateur, qui active lorsqu'il le souhaite la réalisation de l'opération choisie (c'est-à-dire la transformation du modèle en la différence, l'intersection ou l'union du modèle avec l'outil). L'autre utilisation de l'outil effectue en continu cette opération pendant son déplacement.



(a) (b)

FIG. 1.3: (a) exemple de partitionnement d'un polygone, avec l'arbre correspondant, (b) exemple de modèle construit avec le système avec, à gauche, une *évocation* de l'arborescence correspondante (figure extraite de [Nay90]).

Grâce à l'alternance d'un mode *vue* et d'un mode *outil*, l'utilisateur contrôle la position du modèle sculpté ou de l'outil avec une souris 2D. L'article s'intéresse également à des problèmes spécifiques à la représentation arborescente, concernant la réalisation des opérations, et à la gestion du périphérique d'entrée (souris 2D).

Sculpting Polygonal Models using Virtual Tools

J.R. Bill et S.K. Lodha reprennent dans [BL95] l'approche de *sculpture* à l'aide d'*outils virtuels* :

- la **forme** de ces outils est définie par une super-quadrique, ce qui permet d'exploiter deux formulations : implicite pour détecter efficacement les collisions, et paramétrique, pour extraire une forme polygonale, pour l'affichage.
- l'**action** des outils :
  - pousser (gestion des collisions avec l'outil) ou tirer (sommets du modèle contraints sur la surface de l'outil). On précise aussi une fonction d'influence (figure 1.2) qui sert à contrôler la propagation des déformations aux sommets voisins.
  - définir une région de l'objet.

Lorsqu'une région est définie, il est possible de détruire les arêtes/sommets intérieurs (créer des trous), remplir des trous, subdiviser (*augmenter la résolution*), ou *lisser* la région (par subdivisions successives).

L'article insiste aussi sur l'importance de l'interface, et en particulier l'utilité des ombres de l'outil

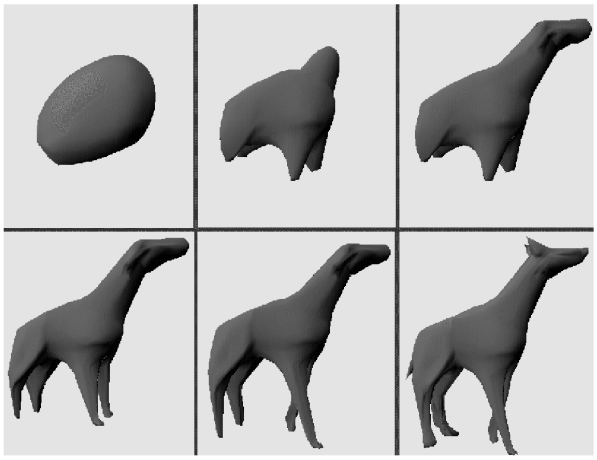


FIG. 1.4: Exemple de figures obtenues avec le système de [BL95]

sur l'objet pour aider au positionnement dans l'espace.

Interactive Mesh Dragging with Adaptive Remeshing Technique

H. Suzuki et al. étudient dans [SSKK98] comment déplacer des éléments d'un maillage. L'idée

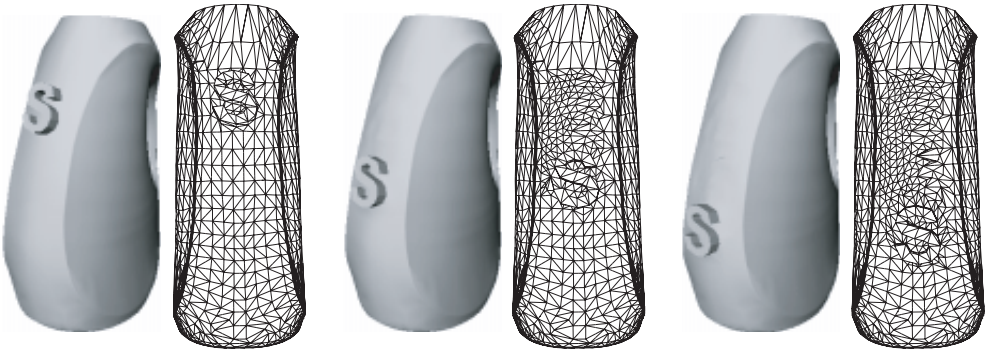


FIG. 1.5: Exemple de translation d'un élément de maillage en forme de S (figure réalisée à partir de [SSKK98]).

est de sélectionner un ensemble de sommets du maillage à déplacer (*sélection*), puis ensuite tout au long de leur déplacement de vérifier la *validité* des triangles adjacents aux sommets frontières de la sélection (*contour*). Les auteurs maintiennent une liste de ces triangles de contour. La validité d'un triangle est basée sur la longueur de ses arêtes et ses trois angles intérieurs. Il est possible de définir des zones correspondant à un triangle bien configuré (valide) et des zones de danger où le triangle devient dégénéré et une modification s'impose (voir Figure 1.6). Les modifications sont *locales* à chaque triangle. Ce sont des opérations simples de partage d'arête (création de sommet), effondrement d'arête, retournement d'arête, qui permettent un traitement rapide. La liste des triangles du contour est main-

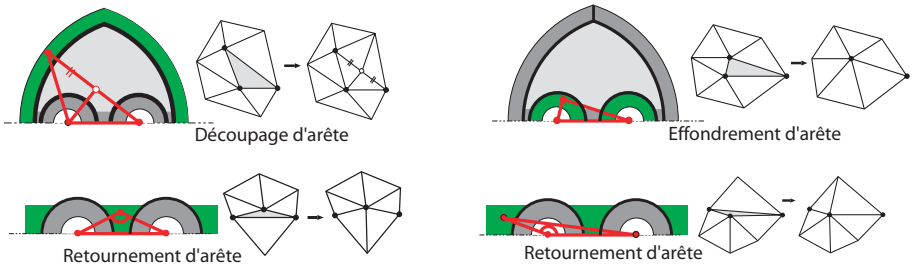


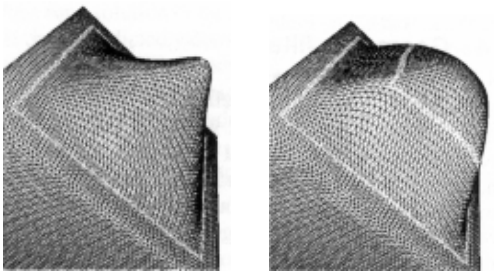
FIG. 1.6: Représentation des zones de danger avec différentes configurations nécessitant une modification. (figure réalisée à partir de [SSKK98]).

tenue lors de ces opérations élémentaires (suppression, insertion des triangles concernés).

Les auteurs relèvent quelques limitations, comme par exemple le fait que les points créés lors de la division d'arêtes sont créés par interpolation. Ainsi, lorsque la sélection traverse une région du maillage, cette dernière est *effacée*, i.e. remplacée par une interpolation du support de la sélection. D'éventuels détails de la surface dans cette région ne sont pas conservés lors de la traversée.

Interactive Multi-Resolution Modeling on Arbitrary Meshes

Du point de vue de l'utilisateur, la démarche proposée par L. Kobbelt dans [KCVS98] consiste à définir une région (c'est-à-dire une suite de *points* connectés par une géodésique, et formant une courbe fermée inscrite sur la surface) qui va limiter l'influence et donner l'échelle des modifications. Ces modifications sont ensuite spécifiées par l'intermédiaire d'une autre courbe inscrite dans



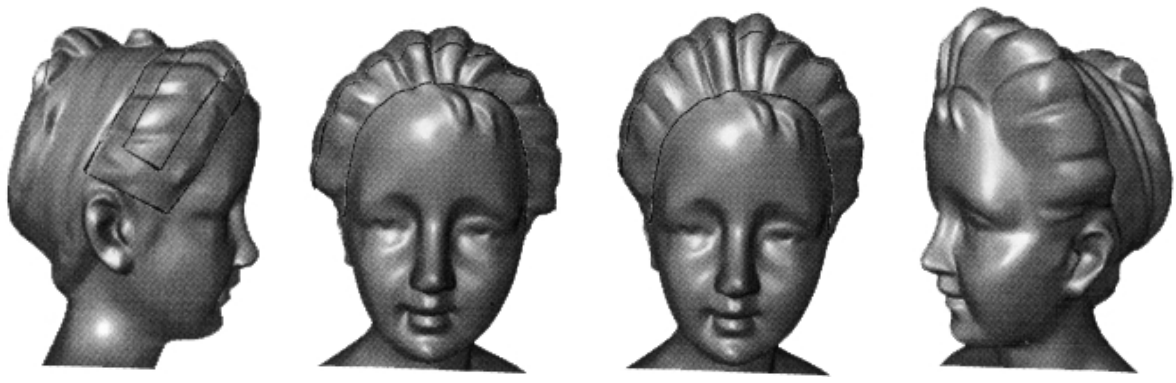
(a) (b)

FIG. 1.7: Déplacement d'un polygone de contrôle à l'intérieur d'une région. La déformation change selon la taille du polygone.

la région, et pas forcément fermée (figure 1.7). Lorsqu'on déplace cette courbe comme un outil, la surface l'interpole toujours de manière C^1 grâce à la minimisation d'une énergie de *tension*(que nous verrons plus en détails dans la section 1.5). L'aspect important est que les détails sont préservés sur la surface modifiée. Les auteurs utilisent pour cela une hiérarchie de simplifications M_i d'un modèle polygonal M_n calculée à partir d'un modèle complexe par simplifications successives. Une étape de la simplification calculant M_{i-1} en fonction de M_i est une suite de :

- effondrements d'arêtes, en interdisant les effondrements d'arêtes adjacentes dans une même suite.
- déplacement de sommets pour *améliorer* la triangulation (augmentation des angles minimum des triangles).

La position stockée du point p de M_i supprimé ou déplacé dans M_{i-1}, est calculée dans un repère local au triangle q de M_{i-1} le plus proche de p : ce calcul n'est pas pénalisant car il est effectué dans une phase de pré-traitement pour construire de la hiérarchie de triangulations (calcul global), puis est maintenu pendant la phase d'interaction (calcul local).



(a) (b) (c) (d)

FIG. 1.8: Exemple de déformation d'un modèle polygonal de buste : (a) définition de la zone de déformation et du polygone de contrôle intérieur, (b) vue de face, (c) déformation par translation du polygone de contrôle, (d) vue de côté du buste déformé.

La technique fonctionne bien, et les auteurs ne gèrent pas la résolution du maillage, car ils travaillent sur des maillages déjà très détaillés. L'objectif visé ici est l'édition de surfaces existantes

plutôt que la construction.

Bilan

Les approches précédentes fournissent des outils intéressants pour éditer ou ajuster des surfaces, mais ne permettent pas réellement d'oublier la représentation polygonale. Il faut même la gérer explicitement dans la plupart des cas : élimination de sommets, augmentation locale de la résolution, gestion délicate de la topologie, ...

On a vu toutefois dans ce contexte l'introduction de la notion d'outil pour abstraire la représentation : en effet, l'utilisateur agit sur la surface à travers l'outil et c'est ce dernier qui traite plus directement avec la représentation.

Les deux dernières contributions ont montré d'autres niveaux d'abstraction, en introduisant des notions d'échelle des modifications pour gérer des détails sur la surface ou la modifier plus globalement sans perdre ces détails. Ces notions sont importantes, car elles autorisent le retour à des modifications plus radicales, sans perdre le travail plus poussé effectué sur les détails, offrant donc plus de liberté dans le processus de construction.

Des travaux sur l'analyse de maillage, liés notament à l'analyse multirésolution, elle-même reliée à la simplification et à la subdivision permettent une abstraction plus grande vis-à-vis des représentations polygonales au prix d'une phase d'analyse globale. Avant de les aborder plus en détail à la section 1.6, nous allons nous intéresser à une autre branche de représentation de surfaces que sont les surfaces Splines, également un des fondements des surfaces de subdvisions.

1.2 Les surfaces splines

Une description générale des surfaces dites *tensor product surfaces* regroupant les carreaux de Bézier, les B-Splines, les NURBS,... est au delà du cadre de ce document. Pour plus de détails, on peut se reporter à [BFK84], ou au livre plus général [FvDFH90].

Notons simplement que ces surfaces largement utilisées dans les outils CAO *classiques* posent différents problèmes pour leur édition. Outre leur manipulation indirecte à travers des points de contrôle, elles sont définies sur des carreaux, ce qui oblige à gérer soigneusement les raccordements, tant pour la continuité que pour l'affichage (problèmes de continuité et de triangulation pour l'affichage). Egalement, pour pouvoir construire des surfaces plus complexes, des courbes de découpe sont utilisées (*trim-curves*), qui compliquent l'affichage et la manipulation.
Une solution potentielle à ces problèmes consiste à recourir plutôt à des patches triangulaires, nous y reviendrons à la fin de cette partie.

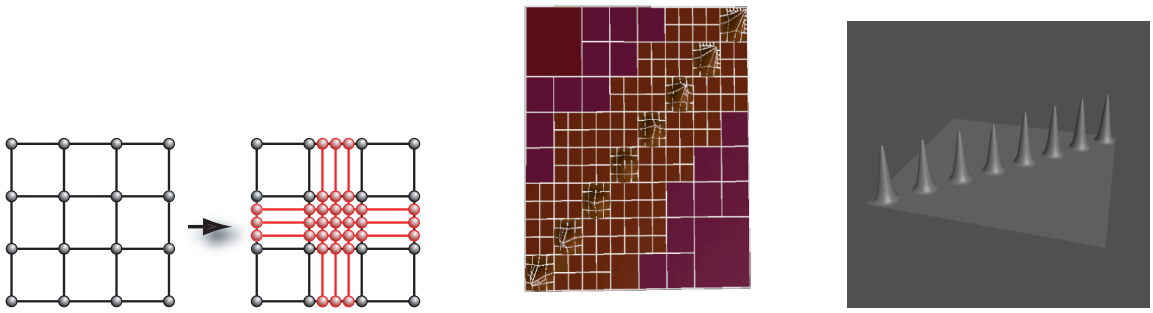
Nous nous intéressons dans ce qui suit à des développements spécifiques venant enrichir les possibilités de déformation / édition des surfaces splines classiques.

Hierarchical B-Spline Refinement

D. Forsey et R. Bartels remarquent dans [FB88] que la manipulation des surfaces *produit tensoriel* peut se révéler extrêmement complexe. Lorsque l'on désire éditer plus finement un carreau, la solution classique consiste à le subdiviser en coupant chaque domaine paramétrique (voir figure 1.9). Le problème est que les domaines paramétriques partagés par les carreaux voisins sont également subdivisés, pour assurer la continuité entre les carreaux, introduisant des points de contrôle inutiles et compliquant la représentation.

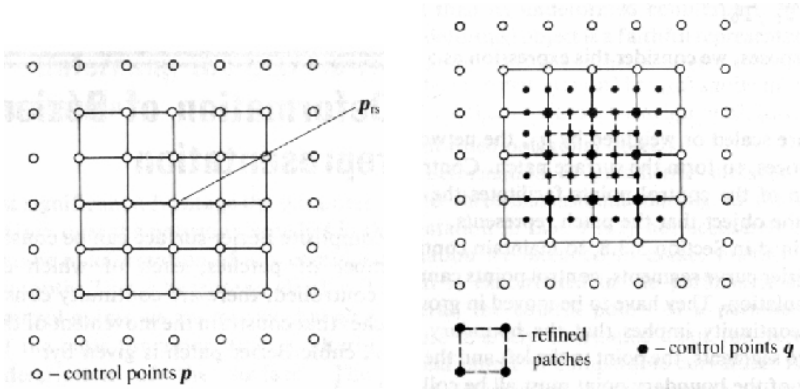
Le premier apport de [FB88] est la définition d'*overlays*. Ce sont des carreaux de spline :

– définis localement (on ne subdivise pas tous les carreaux) et leur extension dépend de la continuité requise.



(a) (b) (c)

FIG. 1.9: (a). Raffinement classique d'un carreaux paramétrique : la subdivision du carreau central entraîne celle des deux domaines de paramètres adjacents. (b) et (c) en subdivisant les carreaux situés sur la diagonale, tous les domaines seraient inutilement subdivisés (cas le pire), ce que les overlays de [FB88] évitent (images extraites de la page web de D. Forsey).



(a) (b)

FIG. 1.10: Raffinement proposé par [FB88] : (a). une surface spline à 16 carreaux avec 49 points de contrôle, (b). les 4 carreaux centraux sont subdivisés en 16 (Figure extraite de [FB88])

– définis comme un déplacement relativement au carreau de spline moins raffiné sous-jacent (au lieu de le remplacer, voir Figures 1.10 et 1.11).

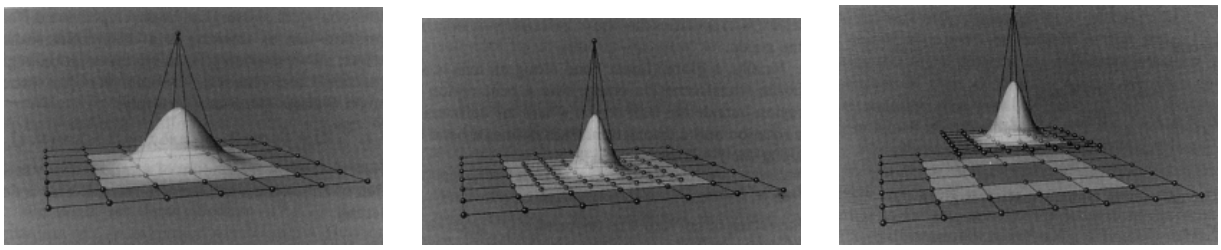
Pour représenter ce déplacement, chaque nouveau point de contrôle q est exprimé dans un repère local centré en $r(u,v)$, point de la surface moins raffinée sous-jacente (surface mère) le plus *influencé* par q, et orienté selon la normale et le plan tangent à la surface en r : $q = r(u,v) + o(u,v)$, avec $o(u,v)$ vecteur de déplacement (*offset*).

Ainsi, un déplacement de q sera pris en compte en modifiant $o(u,v)$, tandis qu'un déplacement de la surface mère se traduira par un déplacement de $r(u,v)$, mettant automatiquement à jour q sans modifier $o(u,v)$: le détail fin *suit* la déformation globale.

Ce procédé de raffinement est répété récursivement et permet à l'utilisateur de modifier le niveau de détail qu'il souhaite, sans perdre les détails éventuellement déjà construits.

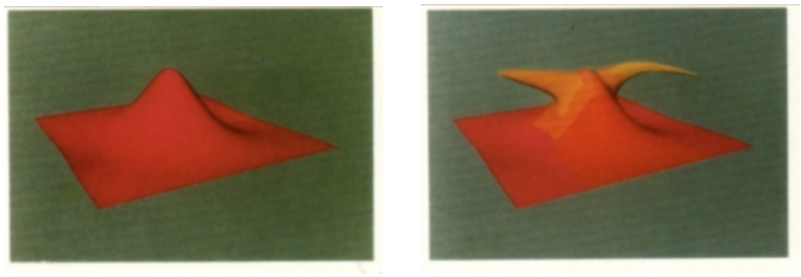
Les auteurs signalent que lorsque deux *overlays* entrent en contact, ils sont fusionnés : les points de contrôle latéraux, de déplacement nul pour assurer la continuité avec la surface mère (Figure 1.11), peuvent alors être déplacés lorsqu'ils deviennent *intérieurs* au nouvel *overlay*.

Un autre apport de ces H-Splines est la manipulation *directe* de la surface à travers des points d'*édition*. Ce sont des points e **prédéfinis** de la surface correspondant à un maximum d'influence d'un des points de contrôle p. Un déplacement de e sera alors répercuté sur le p correspondant. Cela permet de manipuler la surface sans passer par les points de contrôle, ce qui deviendrait vite impossible de par la complexité de la hiérarchie.



(a) (b) (c)

FIG. 1.11: Définition d'un overlay : (a) une surface spline $r(u,v)$ à un niveau de raffinement donné. (b) raffinement des 4 morceaux centraux. (c), les surfaces $r(u,v)$ et $o(u,v)$ séparées, comme elles le sont dans la hiérarchie d'overlays ; on remarque les points de contrôle autour de la surface raffinée destinés à assurer la continuité avec la surface mère (images extraites de [FB88]).



(a) (b)

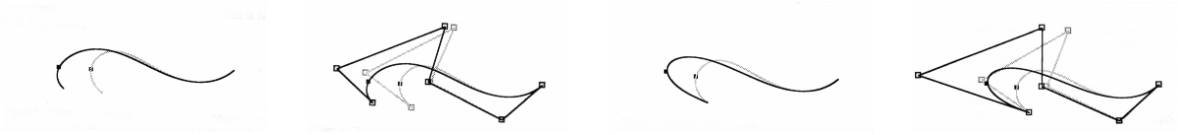
FIG. 1.12: Quelques étapes de modélisation. (a). surface spline à 16 carreaux, 1 point d'édition a été déplacé. (b). après 2 phases de raffinement pour chaque corne (la corne de droite a un raffinement supplémentaire pour réaliser la pointe). Une couleur différente est utilisée pour distinguer un niveau de hiérarchie différent. (images extraites de [FB88])

Cette approche H-Spline multirésolution est très séduisante car elle permet de préciser des détails sur la surface (Figure 1.12, puis de pouvoir revenir à des modifications plus grossières sans les perdre : les détails suivent les modifications.
Toutefois, plusieurs reproches peuvent être formulés envers cette représentation. D. Zorin et al. dans [ZSS97] évoquent par exemple les problèmes posés par la non unicité de la représentation. En effet, il est possible de représenter la même surface de plusieurs manières différentes. Ceci pose problème pour déduire un modèle H-Spline d'une autre représentation, bien que des approches coûteuses aient été proposées (approche de minimsation dans [FW98]). Mais surtout, comme la construction dépend de l'historique des raffinements successifs, il est possible d'aboutir à une représentation difficilement manipulable. D'autres problèmes sont évoqués par C. Gonzalez-Ochoa et J. Peters dans [GOP99], mais nous reviendrons sur cette contribution à la fin de cette section.

A Technique for Direct Manipulation of Spline Curves

B.A. Barsky étudie dans [BB89] l'édition directe de points dans le cas de **courbes** de Bézier et Splines. Il étend toutefois le concept précédent d'édition directe, en ce sens que le point P accessible n'est plus un point d'édition précalculé, mais un point **quelconque**. Il suppose connue la position du point P dans le domaine paramétrique, disons u, étant donnée sa position géométrique, disons (x,y), et explore plusieurs solutions pour déplacer les points de contrôle de manière à ce que la nouvelle courbe interpole la nouvelle position P' du point P choisi : en fixant ou non les points extrémités (Figure 1.13), en ne déplaçant que le point de contrôle le plus influencé par P ou plusieurs points de

contrôle avoisinants (Figure 1.14). Il donne dans chaque cas un jugement qualitatif sur le comportement obtenu pour la courbe.



(a) (b) (c) (d)

FIG. 1.13: Déplacement d'un point d'une courbe : (a) sans fixer les points extrémités (idem avec les points de contrôle en (b)), puis (c) et (d) idem en fixant les points extrémités (Figures extraites de [BB89]).

(a) (b) (c)

FIG. 1.14: Influence du nombre de points de contrôle déplacés : (a) et (b) un seul point de contrôle est modifié ; ce point de contrôle, choisi selon sa *proximité* au point de la courbe sélectionné, n'est pas le même en (a) et (b), bien que celui sélectionné soit *sensiblement* à la même position. (c) courbe obtenue en modifiant les deux points de contrôle du domaine paramétrique où se situe le point sélectionné (Figures extraites de [BB89]).

Geometric Manipulation of Tensor Product Surfaces

B. Fowler essaie dans [Fow92] de rendre plus intuitif le contrôle de carreaux spline. Au libre positionnement d'un point P **quelconque** sélectionné sur la surface, il ajoute la possibilité de spécifier géométriquement ses tangentes. Les tangentes en P sont visualisées à travers un carré, qui sert aussi à les modifier :

- on peut faire pivoter le plan du carré autour de ses axes : ainsi on fixe la direction des tangentes de la surface en P (Figure 1.15.b).
- on peut faire pivoter le plan par rapport à sa normale : simule une torsion (voir 1.15.c).
- en rétrécissant / élargissant le carré, on modifie la norme des tangentes : simule une variation de tension de la surface (voir figure 1.15.d et 1.15.e). En ne modifiant pas le carré uniformément, on peut simuler une tension directionnelle.

L'approche suivante exploite ces techniques d'interaction, mais propose une alternative à la solution hiérarchique par overlays vue précédemment. En effet, comme la décomposition hiérarchique n'est pas unique, il existe plusieurs manières de construire la même surface en utilisant des résolutions différentes, ce qui peut poser problème pour éditer une surface existante au cas où elle serait *mal* construite, par exemple avec plus de raffinements que nécessaires.

Localized-Hierarchy Surface Splines (LeSS)

C. Gonzalez-Ochoa et J. Peters notent dans [GOP99] que la représentation hiérarchique par overlays de [FB88] ne gère que des configurations en *dammier*, c'est-à-dire que les raccords de plus ou moins de quatre surfaces en un point ne sont pas possibles. De plus l'évaluation de la surface par offsets successifs sur les surfaces moins détaillées peut devenir compliquée et instable.

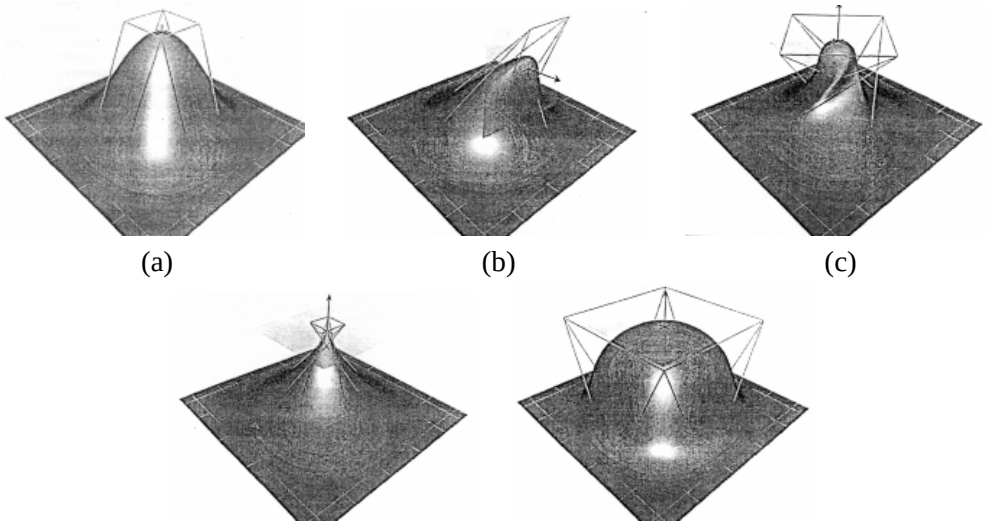


FIG. 1.15: Illustration des concepts de manipulation de surface : (a) positionnement du point sélectionné de la surface, (b) spécification de ses tangentes, et de la torsion (c). (d) et (e) contrôle de la *tension* de la surface (norme des tangentes) à travers la taille du carré (Figures extraites de [Fow92]).

Ils proposent une approche utilisant un maillage comme polygone de contrôle. Les raffinements successifs sont effectués localement sur le maillage, en définissant un sous-maillage stocké comme offset par rapport au maillage précédent. La structure de base est donc une hiérarchie de maillages / éléments de maillage.

La surface elle même n'est pas hiérarchique. Elle est construite avec des triangles de Bézier direc-

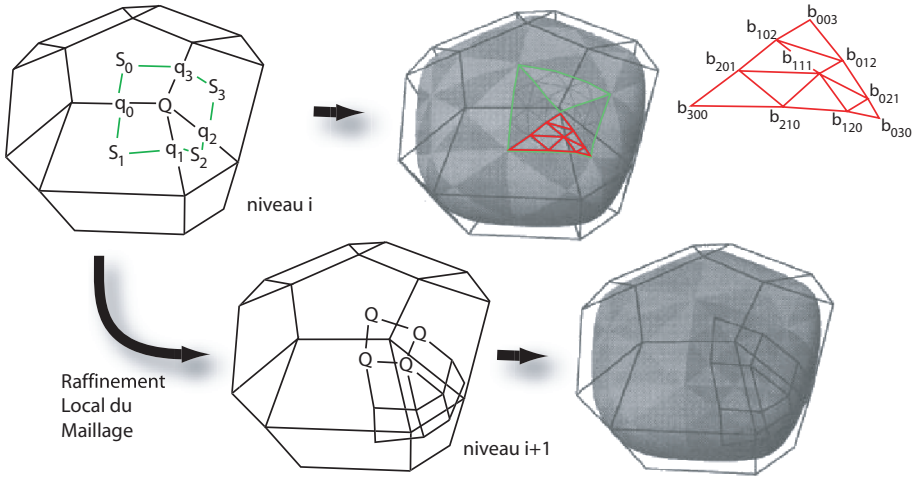


FIG. 1.16: Exemple de maillage. Les neufs points indiqués Q, S_i et q_i définissent quatre patches triangulaires sur la surface à droite (un exemple de patch triangulaire de Bézier est représenté à droite). En dessous, un exemple de raffinement local du maillage, avec la surface correspondante (images extraites de [GOP99]).

tement liés au polygones les plus détaillés de la hiérarchie (en gérant les raccords entre les faces subdivisées ou non). La Figure 1.16 montre un exemple de raffinement local, et les modifications de la surface correspondante.

Un des avantages évidents est que la surface peut être construite à partir d'un maillage initial

quelconque, et en particulier, il est possible durant les opérations d'édition de modifier la topologie du maillage de contrôle, donc de la surface. La figure 1.17 montre le principe et le résultat de modifi-

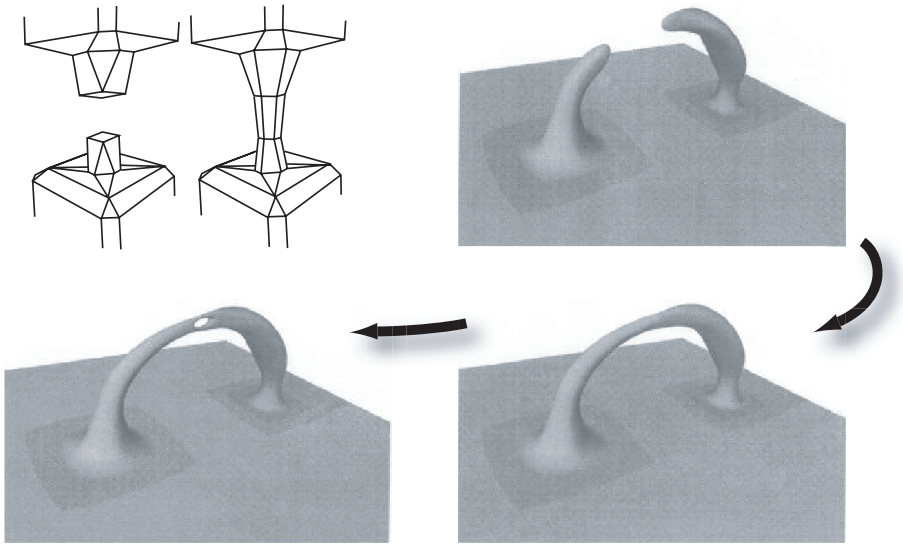


FIG. 1.17: Exemple de modifications de topologie de la surface. En haut à gauche on voit le schéma du principe des modification sur le maillage de contrôle. Ensuite, on voit une séquence pour créer deux protubérances à partir d'une surface plane, puis les connecter et fabriquer une anse, et finalement percer cette dernière (figure réalisée à partir de [GOP99]).

cations de la topologie en cours d'édition.

L'édition de la surface se fait en manipulant directement un point de la surface comme vu précédemment. Il est également possible de contrôler la courbure de la surface le long des arêtes du maillage en modifiant les pondérations utilisées dans le schéma de subdivision lors du raffinement du maillage de contrôle (ces manipulation sont inspirées des schémas de subdivisions que nous aborderons dans la section 1.6).

Bilan

On a vu que la manipulation de surfaces splines peut se faire par interaction directe avec la surface. Les aspects multirésolution qui permettent d'ajouter des détails fins sur la surface, et de les déplacer avec la surface lors de modifications plus importantes, i.e. à une résolution plus grossière, sont très intéressants. Il est même possible de modifier la topologie de la surface en cours d'édition [GOP99]. Toutefois, ces modifications de topologie se font explicitement sur les points du polygone de contrôle, internes à la représentation.

Nous reviendrons sur ces concepts d'édition multirésolution et de gestion de la topologie dans la section concernant les surfaces de subdivision 1.6. Avant cela, nous allons nous intéresser à quelques techniques de déformation indépendantes de la représentation.

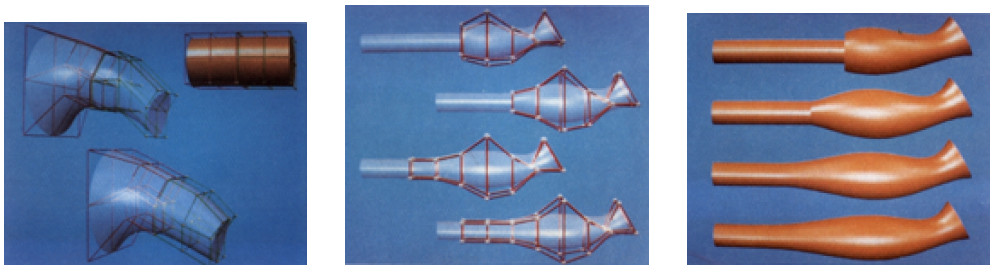
1.3 Déformations indépendantes de la représentation

Les déformations de forme libre, plus connues sous le nom de *FFD* pour *Free Form Deformation* ont été introduites en infographie en 1986 par T.-W. Sederberg et S.-R. Parry dans [SP86].

Free-Form Deformation of Solid Geometric Models

Le principe est de déformer un objet existant en transformant ses coordonnées par une fonction de $\mathbb{R}^3 \longrightarrow \mathbb{R}^3$. De telles transformations avaient déjà été abordées par A.-H. Barr en 1984 dans [Bar84], qui montre quelques cas simulant des torsions, des élongations, des compressions et leurs compositions. T.-W. Sederberg et S.-R. Parry [SP86] proposent plutôt de définir un volume englobant, de le subdiviser éventuellement, puis de le modifier pour altérer en conséquence l'espace, et l'objet intérieur.
Le calcul des coordonnées se fait par extension triparamétrique des splines (avec des polynômes de Bernstein ici, mais les auteurs signalent que la démarche serait similaire avec d'autres bases). Les points définissant le bloc englobant agissent comme des points de contrôle (voir figure 1.18.a).

Ainsi, par extension des propriétés des splines, il est possible de ne déformer que localement l'espace, de contrôler la continuité entre les différentes zones de déformations (par des contraintes sur $k+1$ niveaux de points de contrôles entre deux zones (alignement des points de contrôle de $k+1$ plans des réseaux adjacents, ici), on assure une continuité C^k au raccord (voir figure 1.18)).



(a) (b) (c)

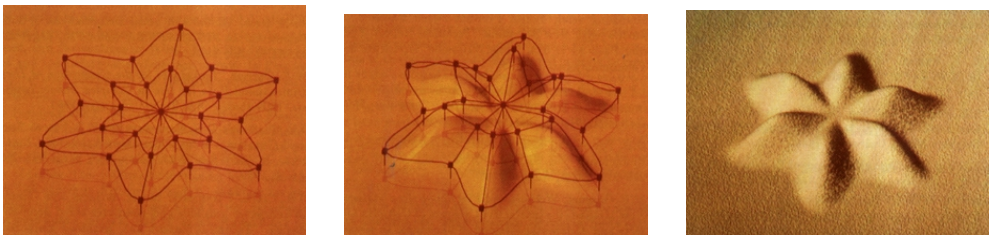
FIG. 1.18: Continuités, discontinuités : (a) : définition d'un réseau de déformation constitué de deux réseaux adjacents ((a) en haut à droite). Déformation C^0 ((a) milieu gauche), obtenue en faisant coïncider les points de contrôle communs. Déformation C^1 ((a) bas droite) obtenue en contraignant un *niveau* supplémentaire de points de contrôle. (b) et (c) : déformations locales discontinuité, continuité C^0, continuité C^1, et C^2, (b) d'une surface avec son réseau de déformation et (c) juste la surface déformée (images extraites de [SP86]).

Les auteurs signalent qu'il est possible d'estimer les variations de volume en intégrant la variation de volume élémentaire (le déterminant de la matrice jacobienne) sur le volume non déformé (supposé plus simple à évaluer) ; ce déterminant étant encadré, de par la formulation avec les polynômes de Bernstein, par le plus grand et le plus petit coefficient, il possible d'encadrer facilement les variations de volume. Les recherches des auteurs les ont amené à étudier les déformations conservant le volume, mais elles sont seulement évoquées dans [SP86].

Extended Free-Form Deformation : A Sculpturing Tool for 3D Geometric Modeling

S. Coquillart remarque dans [Coq90] que les FFD utilisent obligatoirement un parallélépipède comme bloc de contrôle, et propose une extension permettant de spécifier n'importe quelle forme comme polygone de contrôle. Son modèle $EFFD$ pour *Extended Free Form Deformation*, repose sur des blocs prismatiques, et les compose pour spécifier une forme quelconque de réseau de déformation. La modélisation par (E)FFD se déroule en plusieurs étapes :

– **construction d'un réseau de déformation** : consiste à choisir un réseau prédéfini (parallélépipède, cylindre, ...) ou construire des réseaux plus complexes en *reliant* des prismes élémentaires (voir figure 1.19.a).



(a) (b) (c)

FIG. 1.19: Exemple d'utilisation de l'EFFD (images extraites de [Coq90]).

Cette opération de liaison peut se révéler délicate : pour assurer la continuité de la déformation au sein du réseau, il faut contraindre des points de contrôle (les aligner).

- **Associer le réseau à une surface** : pendant cette phase, le réseau est toujours modifiable sans altérer la surface ; on peut l'ajuster pour, par exemple, faire correspondre des points du réseaux à des points de la surface (figure 1.19.a).
- **Geler le réseau** : cela consiste à calculer les positions (u, v, w) par rapport au réseau, des points p de la surface (ou des points de contrôle de la surface). Cela se fait en deux étapes :
  - recherche grossière de l'emplacement du point p (en utilisant la propriété d'enveloppe convexe des volumes de Bézier).
  - puis, calcul des coordonnées (itérativement).
- **Déformer la surface** : les déformations du réseau sont reportées aux points de la surface à partir des nouvelles positions des points de contrôle et des coordonnées (u, v, w) calculées précédemment (figure 1.19.b et 1.19.c).

Free-Form Deformations With Lattices of Arbitrary Topology

R. Mac Cracken et K. I. Joy notent dans [MJ96] que l'approche précédente permet bien de construire des volumes déformants de topologie quelconque, mais la préservation de la continuité entre les volumes élémentaires impose des contraintes sur les points de contrôle internes. Ces contraintes sont complexes à maintenir, et altèrent de plus la flexibilité du contrôle pour l'utilisateur.
Ils proposent dans [MJ96] une formulation analogue en ce qui concerne la structure des volumes déformants (briques élémentaires pour construire un volume de topologie arbitraire) et le mode opératoire (phase de construction du volume, phase de gel pour paramétrer les points du modèle, et phase de déformation). La grande différence est qu'ils utilisent des techniques de subdivisions (proche de celles que nous aborderons dans la section 1.6) au lieu des volumes de Bézier. Le premier avantage qui en découle est que les contraintes de continuités sont gérées automatiquement lors de la subdivision. Le second, est que la subdivision procure plus de stabilité et de rapidité dans la phase de gel.

Direct Manipulation of Free-Form Deformations

W.M. Hsu note dans [HHK92] que, dans ces approches, l'utilisateur doit manipuler des points de contrôle pour spécifier une déformation de la surface sous-jacente, ce qui peut devenir gênant car :

- le nombre de points de contrôle peut rapidement devenir très important.
- la manipulation à travers des points *distants* de la surface peut se révéler peu intuitive (par exemple pour faire un plateau, voir figure 1.20). C'est le cas en particulier si le point de contrôle se trouve à l'intérieur de l'objet (occultation).

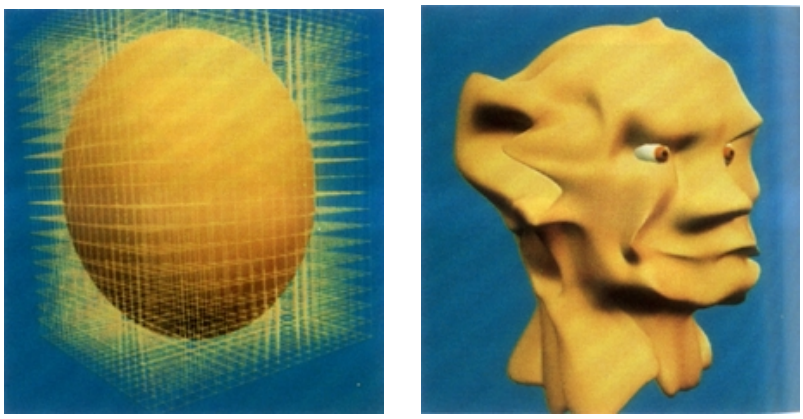
[HHK92] propose plutôt d'éditer directement la déformation par les points de la surface. Il utilise une base B-Spline au lieu des polynômes de Bernstein de [SP86]. Le principal avantage que les auteurs trouvent à cette formulation est la localité du contrôle, mais ils signalent qu'alors la déformation n'est plus forcément incluse dans l'enveloppe convexe des points de contrôle affichés. Les auteurs trouvent

(a) (b)

FIG. 1.20: Positionnement des points de contrôle pour avoir un plateau sur le sommet de la courbe. Le positionnement du point de contrôle central pour former le plateau est peu intuitif (images extraites de [HHK92]).

cela dérangeant dans les cas où le réseau est utilisé explicitement (donc affiché) et leur solution pour retrouver cette propriété consiste à tripler les points de contrôle frontière, garantissant l'inclusion.

Modifier la position d'un point $P(u,v,w)$ de la surface et mettre à jour les points de contrôle pour que la surface déformée passe par P est un problème sous-contraint (une translation de tous les points de contrôle pourrait suffire). Pour faire un choix parmi les solutions possibles, [HHK92] recherche celle qui minimise (au sens des moindres carrés) le déplacement des points de contrôle.
Ce déplacement est estimé à travers les pondérations données par $P(u,v,w)$ et l'équation de la surface $\sum_{i,j,k} P_{i,j,k} B_i(u) B_j(v) B_k(w)$, qui donnent au point de contrôle $P_{i,j,k}$ le poids $B_i(u)B_j(v)B_k(w)$.
La minimisation au sens des moindres carrés n'est pas une recherche itérative et permet donc une mise-à-jour interactive des points de contrôle.
La méthode est étendue au déplacement simultané de plusieurs points de la surface. Le système peut alors devenir sur-contraint (notamment si des points déplacés de la surface partagent des points de contrôle). La solution des moindres carrés aboutit toutefois à une solution cohérente, et les auteurs envisagent même d'utiliser l'erreur trouvée pour diriger un processus de raffinement automatique qui permettrait de revenir à un système sous-contraint. La suite de l'article s'interroge sur les modes d'interaction pour rendre la manipulation des FFD transparente et accentuer la métaphore de sculpture virtuelle. On a retenu l'un des exemple dans la figure 1.21.



(a) (b)

FIG. 1.21: (a) modèle non déformé, visualisé avec le treillis de la FFD. (b) le modèle *sculpté* avec le système proposé, à l'exception des yeux (images extraites de [HHK92]).

Dirichlet Free-Form Deformations and their Application to Hand Simulation

L. Moccozet et P. Kalra jugent limitative l'obligation de définir un **réseau** de déformation (i.e. un ensemble de points et leurs relations). Ils proposent dans [MT97] une autre extension des FFD qu'ils nomment $DFFD$ pour *Dirichlet-FFD*. L'idée est toujours d'exprimer les coordonnées de chaque point P_i de la surface dans un repère dépendant de points de contrôle F_j, mais cette fois-ci les points de

contrôle sont positionnés par l'utilisateur (nombre et position libre). Les déformations sont ensuite calculées comme suit :

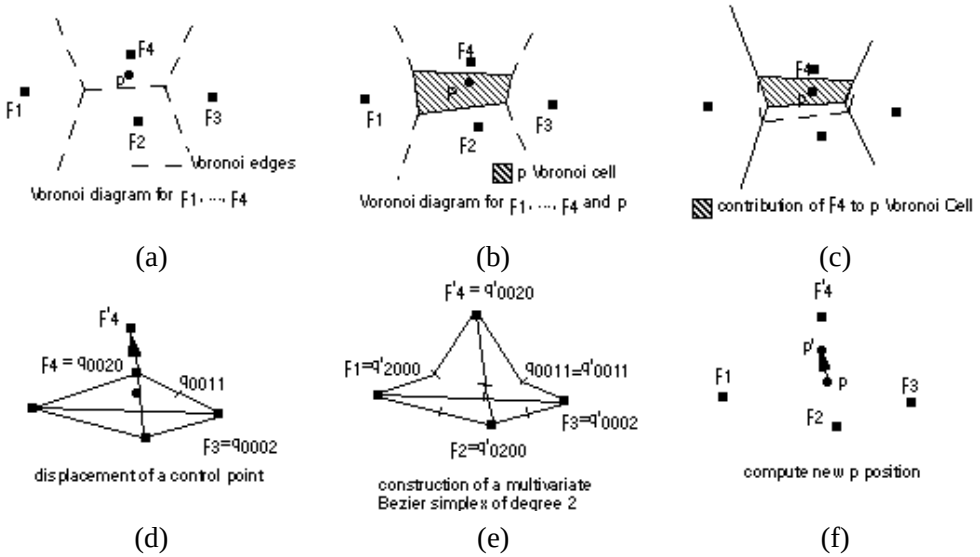


FIG. 1.22: Illustration en 2D de la démarche proposée par [MT97] : (a) diagramme de Voronoï $V(F_j)$ des points de contrôle F_j. (b) pavé de Voronoï de générateurs F_1, F_2, F_3 et F_4 contenant P. (c) *coordonnée* de P par rapport à F_4. (d) déplacement du point de contrôle F_4. (e) construction du simplexe de Bézier associé aux F_j. (f) calcul de la nouvelle position de P (images extraites de [MT97]).

- exprimer chaque point de la surface P_i comme combinaison linéaire des F_j les plus proches, c'est-à-dire les F_j générateurs du pavé de Voronoï qui contient P_i (voir figure 1.22.b).
  Les coefficients de la combinaison linéaire sont non nuls positifs et de somme unitaire ; ils sont appelés les *coordonnées de Sibson*.
  Ils sont obtenus en calculant le volume (resp. l'aire en 2D) de l'intersection entre le pavé généré par insertion de P_i dans le graphe de Voronoï $V(F_j)$ et le pavé de générateurs F_j avant insertion (voir figure 1.22.c).
- les auteurs définissent ensuite un simplexe de Bézier ayant comme points de contrôle les F_j et tel que les coordonnées de Sibson d'un point P soient également ses coordonnées dans le simplexe (voir figure 1.22.e).
- après déplacement des F_j, on re-positionne les P_i grâce à leurs coordonnées dans le simplexe (voir figure 1.22.f).

Les auteurs signalent que la manipulation directe de la surface analogue à [HHK92] peut être obtenue en spécifiant des points de contrôle sur la surface.
Cette méthode couvre une part seulement de l'article, qui poursuit ensuite vers la modélisation des muscles d'une main par cette technique de *DFFD*.

Simple Constrained Deformation for Geometric Modeling and Interactive Design

On pourrait reprocher aux *DFFD* qui déduisent automatiquement les zones d'influence de chaque point de contrôle en fonction des points de contrôles voisins, de ne pas laisser assez de contrôle à l'utilisateur sur ces zones d'influence. L'approche de P. Borrel dans [Bor94], plus ancienne, mais également inspirée des *FFD*, laisse l'utilisateur spécifier un point P sur la surface et un rayon d'influence. Elle est également connue sous le raccourci de *Scodef* pour *Simple constrained deformations*. Le déplacement du point P entraîne aussi celui de ses voisins qui sont à une distance inférieure au rayon d'influence, ce déplacement étant *modulé* par une fonction d'influence B-spline. Cela produit

une bosse en forme de cloche sur la surface. Les zones définies par P et le rayon d'influence peuvent s'intersecter, et conduire à des singularités (contraintes *scodef* non disjointes) compliquant le traitement et la compréhension des déformations générées. P. Borrel inscrit cette approche dans la lignée des FFD, mais certaines déformations sont irréalisables comme la torsion et l'étirement comme le note [Cre99]. Par contre, la localité des déformations et l'absence de *réseau* de déformation permet en principe plus de souplesse / contrôle dans les déformations.

Implicit Free-Form Deformations

C'est à partir de ces dernières remarques que B. Crespin, intéressé par les aspects de pondération des déformations et leur influence locale, propose en 1999 dans [Cre99] une extension de ces concepts. L'idée est d'associer à chaque primitive déformante un repère et une déformation avec une zone d'in-

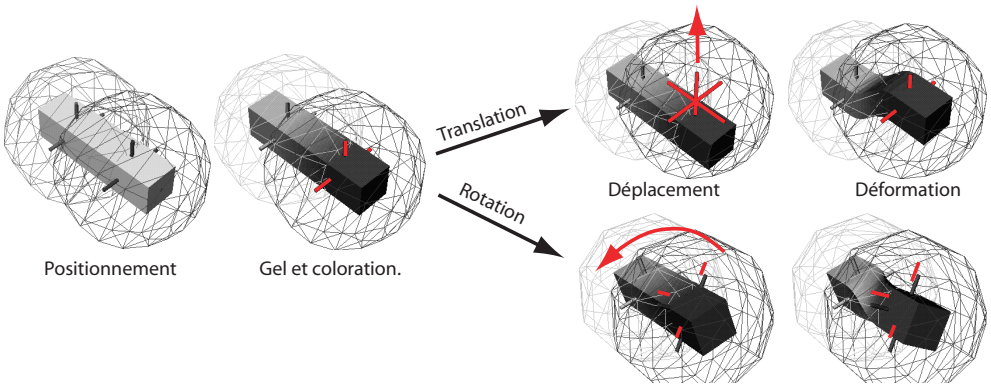


FIG. 1.23: Exemple de déformation globale d'un block par deux primitives $IFFD$ (images provenant de [Cre99]).

fluence pondérée par un potentiel scalaire, d'où l'analogie avec les surfaces implicites qui correspondent à une *tranche-iso* d'un champ scalaire $3D$ (voir section 1.7). B. Crespin propose deux types de déformations qu'il appelle *globales* et *locales*, seules les dernières prenant en considération la gestion de la continuité entre les zones déformées et les zones non déformées.

Bilan

Les techniques de FFD permettent une abstraction du modèle décrivant la surface, et donc potentiellement une généricité maximale vis-à-vis de la représentation.
Notons d'abord que du fait de cette indépendance, il faut toujours disposer d'un modèle initial : pas de création *from scratch*.
Remarquons aussi que le modèle initial doit être finement défini. En effet, dans le cas de fortes déformations, si le modèle par exemple polygonal, n'est pas suffisament finement triangulé, des *étirements* peuvent apparaître ; c'est-à-dire que les polygones représentant la surface initiale peuvent devenir visibles. Ces problèmes de résolution des objets déformés ne sont pas guidés par ces techniques, donc délicats à arranger. Des extensions ont été proposées dans cette voie, comme par exemple les travaux de thèse de P. Decaudin [Dec96] ou plus récemment J. Gain qui propose des techniques pour gérer la résolution du maillage par des techniques de subdivision (voir section 1.6) et décimation dans [GD99].
Selon la complexité du modèle, les déformations peuvent devenir compliquées. Par exemple dans le cas de la Figure 1.24.a, si le volume est automatiquement construit, il est impossible de séparer les deux branches (elles sont déformées ensemble). La construction explicite d'un volume de déformation peut aussi s'avérer complexe / pénible.
Remarquons également que peu des techniques présentées permettent la manipulation directe de la

surface. On se retrouve alors dans la situation de manipuler des points de contrôles hors de la surface. Les dernières techniques présentées avancent le placement de ces points de contrôles sur la surface pour fournir la sensation de la manipuler directement. Dans ce dernier cas, les déformations obtenues sont par ailleurs assez proches des déformations directes sur les maillages avec des fonctions d'influences (voir section 1.1), l'interêt de ces alternatives plus complexes diminue donc d'autant.

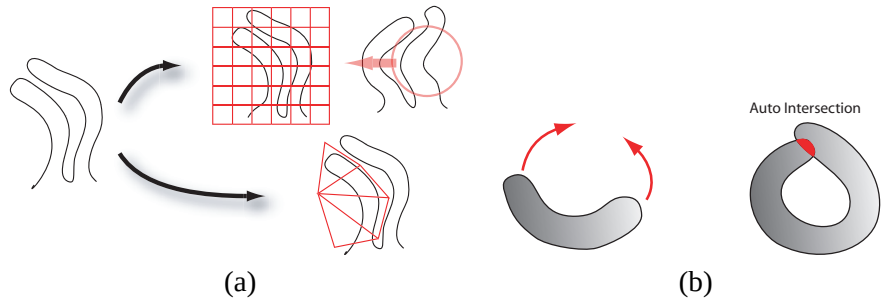


FIG. 1.24: (a) Illustration des problèmes de déformations sur un modèle *compliqué* et des difficultés éventuelles de la construction d'un volume de déformation. (b) Illustration des problèmes d'auto-intersection.

Enfin, le plus gênant en ce qui nous concerne ici est que ces techniques ne gèrent pas les changement de topologie. Il est impossible de creuser un trou ou de séparer / recoller deux éléments. De plus les déformations successives peuvent amener la surface à s'auto-intersecter (voir Figure 1.24.b). Rien ne permet de détecter ou traiter ces situations *extrêmes* ici, mais courantes lors de la construction d'une forme dont on ne connaît pas toutes les caractéristiques au départ.

D'autres techniques pour modifier ou générer / calculer des surfaces sans entrer directement dans la représentation géométrique sous-jacente sont issues des *modèles physiques*. Nous allons faire une distinction dans cette branche entre les modèles générateurs de comportements qui calculent l'évolution d'objets dans le temps, et les modèles générateurs de surfaces dont le but est de calculer des surfaces *optimales* selon certaines contraintes.

1.4 Modélisation à l'aide de modèles physiques

Le but initial des méthodes par modèles physiques était d'aider à la synthèse d'animations réalistes. La description qui en est faite ici est succinte et ne concerne que des articles assez anciens. En effet, l'idée d'utiliser les modèles physiques pour la sculpture interactive a été abandonnée assez rapidement au début de cette thèse. Les articles cités montrent cependant les principes de simulation de matériaux plastiques et constituent une bonne introduction aux surfaces d'optimisation que nous aborderons en 1.5.

Elastically Deformable Models

D. Terzopoulos et al. présentent dans [TPBF87] un modèle dynamique simplifié d'élasticité pour des courbes, des surfaces ou des solides. Ce modèle utilise une énergie de contraintes associée aux déformations, et prend en compte des forces extérieures comme la gravité, une tension (avec un ressort), un contact avec un fluide visqueux ou une collision. L'article détaille également les schémas de différences finies employés pour l'intégration le long d'une surface et dans le temps.

Ce type de modèle élastique ne saurait convenir dans un cadre de sculpture / modelage, puisqu'une fois les forces disparues, la surface retrouve son état initial.

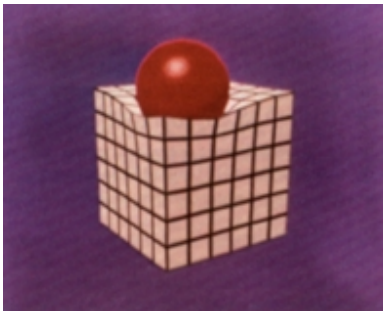


FIG. 1.25: Exemple de comportement élastique pour un cube lors d'une collision avec une balle rigide [TPBF87]

Modeling Inelastic Deformations : Viscoelasticity, Plasticity, Fracture

D. Terzopoulos et K. Fleischer étendent dans [TF88] le précédent modèle pour simuler des comportements inélastiques. Toujours avec une énergie de contraintes, obtenue par exemple pour une surface par :

$$\begin{aligned}\delta_e \varepsilon(e) &= w_{00}e - \frac{\partial}{\partial u_1}\left(w_{10}\frac{\partial e}{\partial u_1}\right) - \frac{\partial}{\partial u_2}\left(w_{01}\frac{\partial e}{\partial u_2}\right) \\ &\quad + \frac{\partial^2}{\partial u_1^2}\left(w_{20}\frac{\partial^2 e}{\partial u_1^2}\right) + 2\frac{\partial^2}{\partial u_1 \partial u_2}\left(w_{20}\frac{\partial^2 e}{\partial u_1 \partial u_2}\right) + \frac{\partial^2}{\partial u_2^2}\left(w_{02}\frac{\partial^2 e}{\partial u_2^2}\right)\end{aligned}$$

où e est la déformation par rapport à la forme de référence (figure 1.26) et les $w_{i,j}$ sont des pondérations traduisant les propriétés physiques du matériau simulé.

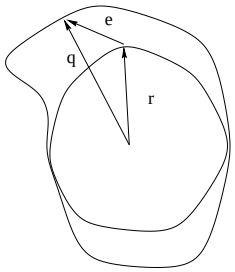


FIG. 1.26: r est la forme de référence, du solide non déformé, q est la forme déformée, et e est la déformation : différence entre r et q.

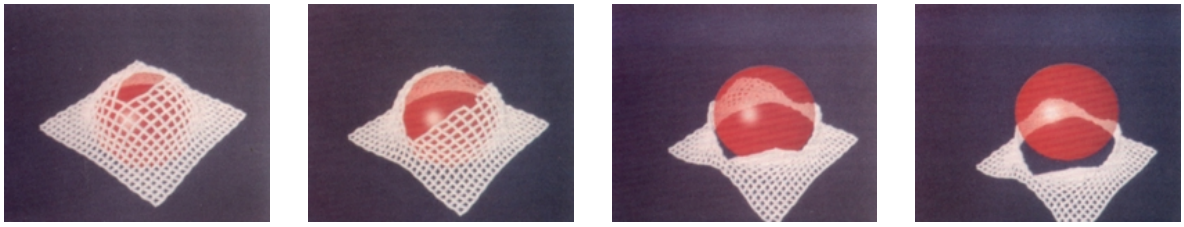
Les comportements inélastiques sont simulés en faisant évoluer la forme de référence :

– comportement visqueux : à chaque pas de temps, on ajoute à r une fraction de la déformation e

$$\dot{r}(u,t) = \frac{1}{\eta(u)}e(u,t)$$

– comportement plastique : quand la déformation ou les contraintes internes dépassent localement un seuil donné, la forme de référence absorbe une partie de la déformation, et les propriétés du matériau (les w_{ij}) sont modifiées.

– fracture : quand les contraintes ou les déformations dépassent un seuil de fracture, les coefficients w_{ij} sont annulés localement aux points où la déformation (et les contraintes) est la plus forte (voir figure 1.27).



(a) (b) (c) (d)

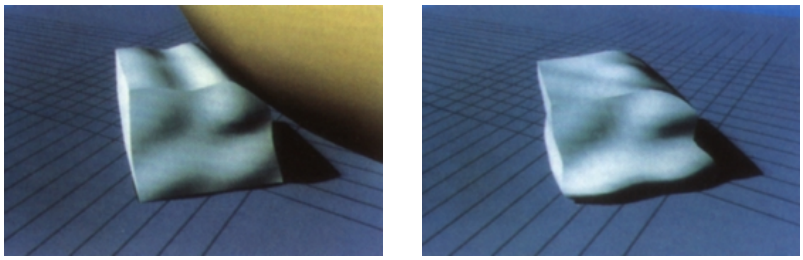
FIG. 1.27: Exemple de fracture simulée par [TF88].

Constraint Methods for Flexible Models

J.C. Platt et A.H. Barr proposent dans [PB88] deux nouvelles méthodes basée sur des contraintes. La première, appelée *RC* pour *Reaction constraints*, est un moyen rapide et simple applicable à une **masse ponctuelle** qui doit satisfaire **une** contrainte. Elle consiste à ne retenir que la partie compatible avec la contrainte d'une force F_{in} (projection dans l'espace non contraint). Les auteurs explicitent cette formulation dans le cas de particules devant suivre un chemin fixé ou rester sur un plan.

La deuxième, appelée *ALC* pour *Augmented Lagrangian Contraints* est un mélange entre :
- les méthodes de pénalité : pour minimiser une énergie potentielle $f(x)$ d'un objet sous des contraintes $g(x) = 0$, x décrivant l'état de l'objet, on va minimiser $f(x) + c(g(x))^2$, où c est une constante. Plus c est grande, plus la contrainte sera *satisfaite* : c'est cette inexactitude qui fait la force et la faiblesse des méthodes de pénalité.
- les méthodes de *contraintes de Lagrange* : on rajoute une variable λ par contrainte, et on minimise alors $f(x) + \lambda g(x)$, par une méthode modifiée de descente du gradient discutée dans l'article, qui consiste à évaluer le gradient avec :

$$\begin{aligned} \dot{x} &= -\frac{\partial f}{\partial x} - \lambda \frac{\partial g}{\partial x} \\ \dot{\lambda} &= +g(x) \end{aligned}$$



(a) (b)

FIG. 1.28: Exemple de comportement plastique simulé par [PB88]

Cette méthode satisfait exactement les contraintes, et les auteurs exposent son utilisation pour simuler des solides élastiques, malléables, plastiques tout en maintenant leur volume constant (exemple de simulation obtenue figure 1.28).

Heating and Melting Deformable Models (From Goop to Glop)

D. Terzopoulos et al. ajoutent à une modélisation similaire aux déformations élastiques, la simulation de la diffusion de la chaleur au sein du matériau dans [TPF89].
La matière simulée est discrétisée en un réseau hexaédrique de masses-ressorts. A chaque point est

associée une masse et une température. Les ressorts ont une raideur et une longueur initiales fixées en fonction des propriétés élastiques et une *résistance thermique par unité de longueur*. La raideur au-delà d'un certain seuil de température moyenne (entre les deux extrémités du ressort) diminue proportionnellement à l'augmentation de température, jusqu'à s'annuler.

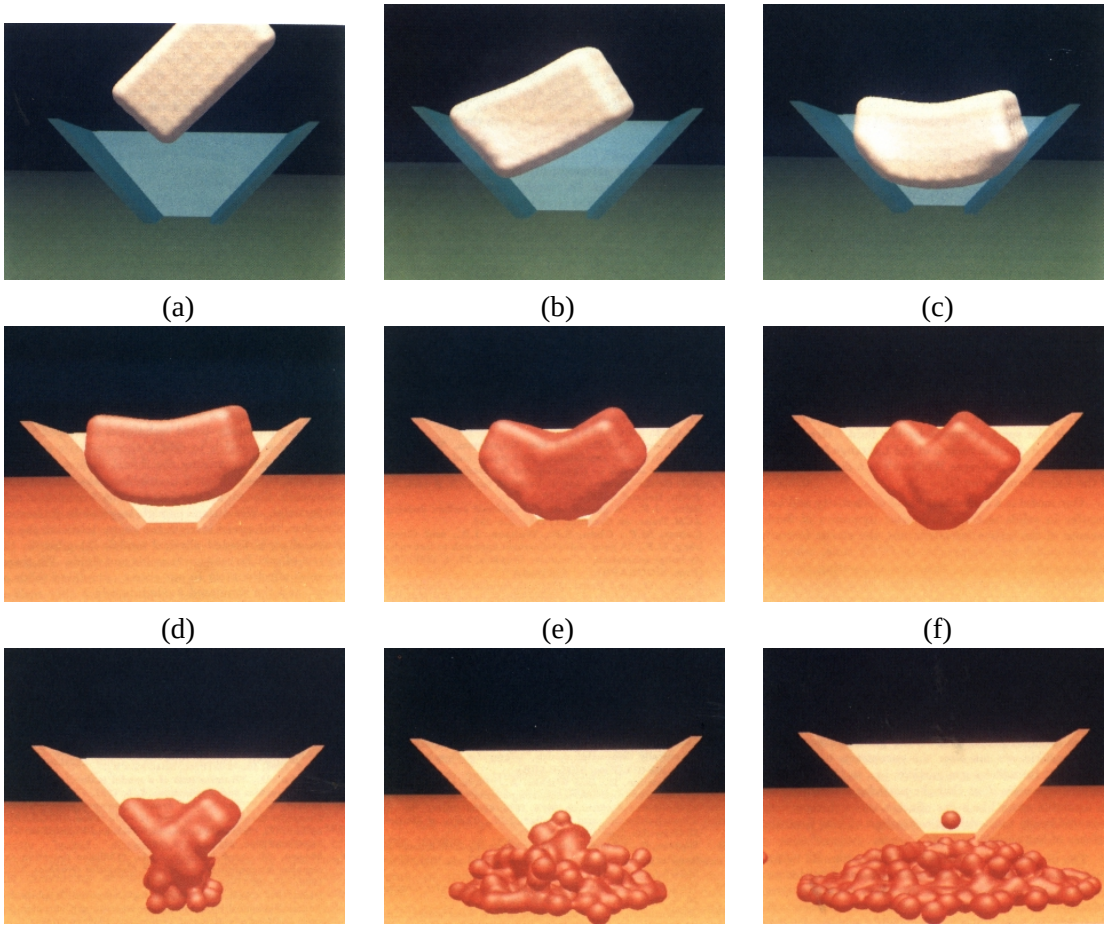


FIG. 1.29: Exemple de la chute et fonte d'un objet dans un cône *chauffant* : (a) chute, (b) collision, (c) arrêt dans le cône froid (avec les frictions). Entre (c) et (d), le cône élève sa température, visualisée par les couleurs rougeoyantes qui diffusent dans l'objet. Ce dernier se ramollit en (e) et commence à fondre en (f), pour se liquéfier finalement et passer à travers l'ouverture au fond du cône progressivement (g), (h) et (i) (images extraites de [TPF89]).

Si une particule se trouve avec tous ses ressorts de raideur nulle, elle est déconnectée du solide déformable et son interaction avec les autres particules est alors régie par une fonction potentielle de la distance de type Lennard-Jones (répulsion à courte distance, attraction à longue distance).
Les auteurs s'intéressent également aux collisions de ces particules avec un obstacle polyhédrique et utilisent pour cela la méthode des *Reaction Constraints* de [PB88] en ajoutant une force de friction *pour plus de réalisme* (figure 1.29.b et 1.29.c).
Comme illustration, l'article montre des images extraites d'une animation où un bloc déformable tombe dans un cône froid (à 0^oC)(figure 1.29.a à 1.29.c), puis en réchauffant le cône (jusqu'à 10^oC), le matériau se liquéfie (au delà de 8^oC) (figure 1.29.d à 1.29.f)et passe à travers l'ouverture du cône (figure 1.29.g à 1.29.i).

Bilan

Les modèles physiques sont très intéressant car ils simulent le comportement physique d'une forme constituée d'un matériau donné. Dans le cadre de leur utilisation pour la sculpture virtuelle, les avantages seraient de produire un comportement *intuitif*, car correspondant à un matériau réel connu dans le cadre d'une interaction de type sculpture. Dans une situation imitant par exemple la poterie, cela pourrait laisser le choix à l'utilisateur d'utiliser de l'argile plus ou moins sèche (*grasse* ou *maigre*). L'intérêt ici serait la rapidité du changement en évitant le délais de séchage ou le travail de ré-hydratatation.
Les temps de calcul interdisaient tout espoir d'utilisation interactive lorsque cette thèse a débuté fin 97. Notons que le temps réel est maintenant possible pour des modèles élastiques non fracturables [Deb00, DDCB01]. A l'heure actuelle, il n'existe toujours pas de simulation temps réel par modèles physiques pour des matériaux plastiques, et à fortiori fracturables. Ces modèles n'ont pas été étudiés plus en détail ici, mais ils introduisent les principes sur lesquels reposent les surfaces d'optimisation.

1.5 Les surfaces d'optimisation

Les surfaces *d'optimisation* sont des surfaces qui minimisent une énergie. Dans la plupart des cas, l'énergie de la surface est liée à sa tension et sa torsion et sert juste à lui assurer de *bonnes* propriétés (surfaces dites *minimales*). Dans cette catégorie, on range les approches connues comme *variationnal surface* ou *surface fairing* et plus récemment *discrete fairing* directement sur des maillages.

Deformable Curve and Surface Finite-Elements for Free-Form Shape Design

G. Celniker et D. Gossard [CG91] proposent de construire un objet en plusieurs étapes :
– définir des courbes caractéristiques en 3D, qui définissent le support de la surface et les singularités (discontinuités comme des bords, des arêtes, ...)
– envelopper ces lignes par une surface minimisant un critère de qualité de la surface.
– modifier la surface obtenue en appliquant des *charges* ou des *forces* tout en maintenant les contraintes de points, tangentes, normales ou courbes spécifiées par l'utilisateur.
Le critère de qualité utilisé mesure la déformation de la surface à travers une énergie similaire à celle exposée dans [TPBF87] :

$$E_{\text{déformation}} = \int_{surface} (\alpha E_{\text{élongation}} + \beta E_{\text{torsion}})$$

qui s'exprime plutôt sous forme différentielle. Les auteurs ajoutent également une partie dynamique (dépendant du temps) et l'équation devient :

$$\begin{aligned} f(w,t) \quad &= \frac{\partial}{\partial t}\left(\rho \frac{\partial w}{\partial t}\right) + \mu \frac{\partial w}{\partial t} + \\ &\left[\left(\frac{\partial^2(\beta_{11} w_{uu})}{\partial u^2} + \frac{\partial^2(\beta_{12} w_{uv})}{\partial u \partial v} + \frac{\partial^2(\beta_{22} w_{vv})}{\partial v^2}\right) - \left(\frac{\partial(\alpha_{11} w_u + \alpha_{21} w_v)}{\partial u} + \frac{\partial(\alpha_{12} w_u + \alpha_{22} w_v)}{\partial v}\right)\right] w \end{aligned}$$

avec t le temps, $w = w(u,v,t)$ un point de paramètres (u,v) de la surface au temps t, et $f(w,t)$ les *forces* appliquées par l'utilisateur en ce point, ρ la densité de masse, μ un terme de viscosité. Les paramètres β_{ij} et α_{ij} sont les coefficients de torsion et d'élongation, et w_u, w_{uv} sont des raccourcis pour $\frac{\partial w}{\partial u}$ et $\frac{\partial^2 w}{\partial u \partial v}$.
La solution est calculée par éléments finis (composition de fonctions de base sur des éléments triangulaires) pour conserver la continuité de la surface, et par différences finies pour l'intégration selon le temps. Une discussion des auteurs sur les *énergies des dérivées des différents ordres* précise que les surfaces tendront à être C^3, bien que les fonctions de base ne soient que C^1.
Pour ce qui est de l'interface, l'article propose de doser l'intensité des forces appliquées avec un

ascenseur (*slider*), mais ne précise pas la manière dont les courbes caractéristiques de départ, ou d'autres paramètres comme les pondérations α et β ou les directions/points d'application des forces sont spécifiées.

Linear Constraints for Deformable B-Spline Surfaces

Dans cette approche [CW92], G. Celniker et W. Welch s'intéressent cette fois à une surface *B*-spline. L'article propose une approche similaire à [CG91] en distinguant deux types d'outils pour modifier la surface :

- les outils de sculpture : en appliquant des forces, des pressions, des tensions (avec des *ressorts*) dont les influences sont prises en compte de manière similaire à [CG91] en résolvant :

$$M\ddot{x} + B\dot{x} + K_\sigma x = f_\sigma(w,t)$$

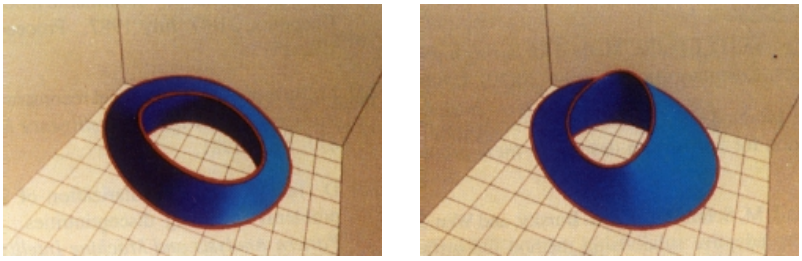
  où $w(u,v)$ décrit la surface dont les points de contrôle sont $x = [P_{0,0}, P_{0,1}, ..., P_{m,n}]$, f_σ est la force appliquée, $M = \rho Id$ avec ρ la densité de matière, $B = \mu I_d$ avec μ une viscosité stabilisant l'équation et $K_\sigma = \int_\sigma (\alpha E_{elongation} + \beta E_{torsion})\, dudv$ est la *matrice de raideur*
- des contraintes géométriques : ce sont des points avec des tangentes ou une normale, qui peuvent être fixés, ou évoluer dans le temps, mais en restant toutefois fixe dans le domaine paramétrique (u,v), sinon les contraintes ne peuvent plus s'exprimer linéairement. L'article expose également une méthode pour contraindre les tangentes ou les normales le long d'une courbe (fixe dans le domaine de paramètres également) en minimisant une erreur.

Variationnal Surface Modeling

W. Welch and A. Witkin étendent dans [WW92] cette formulation en subdivisant les *B*-splines, offrant ainsi la sensation d'une surface malléable sans limite apparente.
L'article s'inspire des overlays de [FB88] en les simplifiant : ici, on somme simplement les contributions au lieu de définir des décalages (*offset*) dans un repère local à la surface pour conserver la linéarité de la formulation.
Classiquement, la surface évolue (à travers ses points de contrôle) pour minimiser une fonction objectif reliée à l'aire et la courbure principale. Comme ici il n'est pas question de forces et d'intégration dans le temps, la contrainte pour éviter l'écrasement de la surface (car tous les points de contrôle confondus est une solution optimale !), la fonction objectif minimise également le déplacement des points de contrôle.
La surface est manipulée par le biais des points ou courbes de contraintes spécifiés : ces derniers sont déplaçables et interpolés par la surface (voir figure 1.30). Toutefois, ils ne sont pas déplaçables dans le domaine paramétrique, c'est-à-dire qu'on n'a pas d'effet de glissement des caractéristiques sur la surface (comme figure 1.38.c).

Functional Optimization for Fair Surface Design

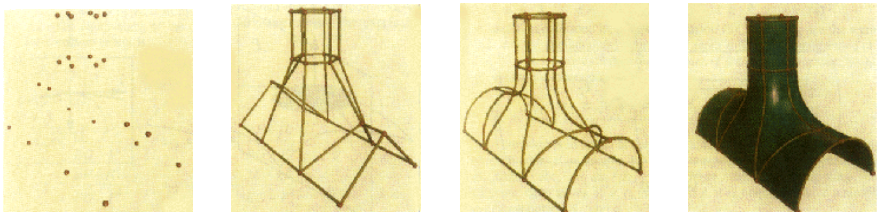
L'approche proposée par H.P. Moreton et C.H. Séquin dans [MS92] diffère des précédentes sur plusieurs points. D'abord la fonction estimant la *qualité* de la surface considère la **variation** de courbure, plutôt que la courbure. Les auteurs justifient ce choix par la recherche de forme *simples* : en effet, dans le cadre de la minimisation des courbures, les cyclides (cylindres, sphères, cônes, tores) ne sont pas des surfaces optimales.
Egalement, la démarche proposée pour définir une surface est différente :



(a) (b)

FIG. 1.30: Exemple de déplacement d'une courbe de contrainte entre deux surfaces : la courbe intersection est interpolée (figure extraite de [WW92]).

– d'abord, on définit un **réseau** de contraintes, destiné à servir de support aux morceaux à trois ou quatre côtés utilisés pour construire la surface. Ce réseau est une suite de segments reliant des points, et définissant la connectivité des carreaux (voir figures 1.31.a et 1.31.b).
– dans une seconde phase, ce réseau de segments est optimisé en un réseau de courbes se raccordant de manière G^2, avec une variation de courbure minimale (voir figure 1.31.c). Ce calcul est non linéaire (méthode itérative de descente du gradient).
– ensuite, des carreaux de Bézier sont définis avec ces courbes comme support. La minimisation des variations de courbure est également recherchée de manière itérative.
La continuité des tangentes entre les morceaux est assurée de manière *souple* (i.e. non stricte) par une méthode de pénalité (un terme ajouté à la fonction objectif qu'on minimise). Les auteurs justifiant leur démarche à posteriori, la continuité obtenue étant suffisante sur leurs exemples (voir figure 1.31.d).



(a) (b) (c) (d)

FIG. 1.31: Processus de modélisation proposé dans [MS92] : (a) définition des points caractéristiques du modèle, (b) définition du réseau définissant la surface entre ces points. (c) le réseau de courbes à variation de courbure minimale calculé. (d) la surface obtenue.

Ces techniques sont basées sur des méthodes non-linéaires, et sont loin de l'interactivité, mais les surfaces obtenues et leur construction par réseau de contraintes sont assez originales.

Surface Modeling with Oriented Particle Systems

R. Szeliski et D. Tonnesen dans [ST92] jugent très limitative l'utilisation de morceaux de splines dans le cadre d'un modeleur interactif : cela demande de connaître à l'avance la forme générale de l'objet qu'on veut modeler pour pouvoir fixer la connectivité des domaines paramétriques (et la topologie de l'objet).
Les auteurs proposent de décrire plutôt la surface avec des particules orientées. Ce sont des particules interagissantes, munies d'un repère local. Plusieurs termes régissent leur comportement (en particulier, leur arrangement surfacique) et sont expliqués qualitativement :

– $\Phi_p(n_i, r_j) = (n_i.r_{ij})^2\Psi(\|r_{ij}\|)$ terme de coplanarité, qui est minimum quand la normale n_i à la particule surfacique i est orthogonale à r_{ij} le vecteur *distance* entre i et la particule j.

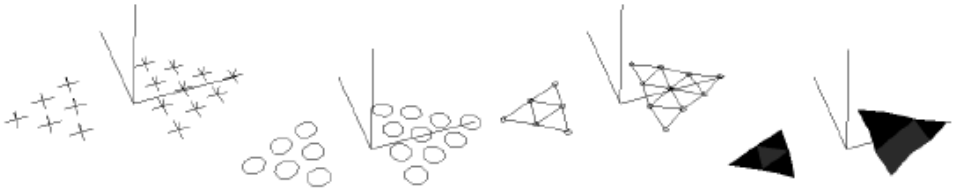


FIG. 1.32: Différentes représentations des particules orientées (figure extraite de [ST92]).

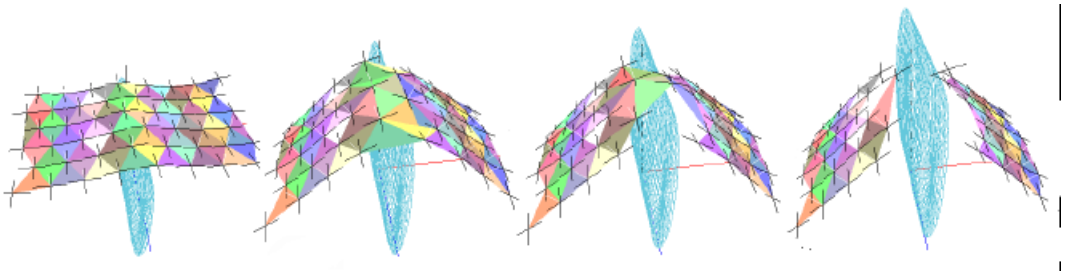


FIG. 1.33: Exemple de fracture : un outil coupe la surface en deux morceaux (figure extraite de [ST92]).

- $\Phi_n(n_i, n_j, r_{ij}) = \|n_i - n_j\|^2 \Psi(\|r_{ij}\|)$ terme de conormalité pour limiter les torsions, qui est minimum quand les normales aux particules surfaciques i et j sont colinéaires.
- $\Phi_c(n_i, n_j, r_{ij}) = ((n_i + n_j).r_{ij})^2 \Psi(\|r_{ij}\|)$ terme de cocircularité, minimum quand les normales sont antisymétriques par rapport à r_{ij}.
- $\Phi_{LJ}(r_{ij})$ un terme d'interaction de type Lennard-Jones (attraction à longue distance, répulsion à courte distance).

Ainsi, l'énergie $E_{ij} = (\alpha\Phi_{LJ} + \beta\Phi_p + \gamma\Phi_n + \lambda\Phi_c)$ régit le comportement des particules, via les forces en dérivant : $f_{ij} = \nabla E_{ij}$.
La particule i reçoit ainsi les forces (bilan des forces) $f_i = \sum_j f_{ij} + f_{ext} - \beta_0 v_i$ (idem pour les couples, bilan des moments). Son mouvement est calculé par intégration (méthode d'Euler) et conduit itérativement à la minimisation de $\sum_{ij} E_{ij}$ pendant la modélisation.
L'interaction avec la surface est également abordée :

- interaction directement avec les particules : ajouter, détruire, déplacer une particule est possible.
- interaction à travers des *outils* dont la **forme** peut être un plan, une sphère, un cylindre, ou un polyèdre. Cet outil peut servir à :
  - repousser les particules (outil solide)
  - attirer les particules (coller à sa surface, ou les aspirer dans son intérieur)
  - gommer des particules.

Ainsi, les surfaces peuvent être coupées, collées. On peut également créer des discontinuités en *désactivant* certains termes d'énergies (voir figure 1.33). La surface peut aussi être étendue ou réduite en ajoutant/supprimant des particules, ce que les auteurs proposent d'évoquer par la métaphore d'un outil *chauffant* (voir figures 1.34 et 1.35).
L'article propose aussi plusieurs modes de rendu de ces particules surfaciques : avec des croix, des disques, ou bien par une triangulation de la surface, mais ce dernier mode n'est pas accessible interactivement.

Free-Form Shape Design Using Triangulated Surfaces

C'est ce problème d'absence de surface que W. Welch et A. Witkin abordent dans [WW94]. Ici la surface est décrite par des points d'échantillonnage et une triangulation associée, formant un *maillage* maintenu, c'est-à-dire modifié localement, pendant l'interaction.

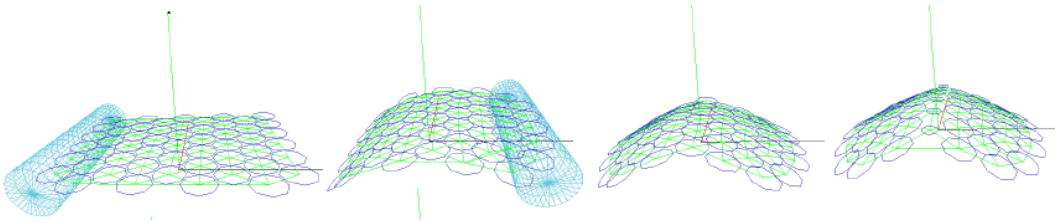


FIG. 1.34: Exemple de création de discontinuité (figure extraite de [ST92]).

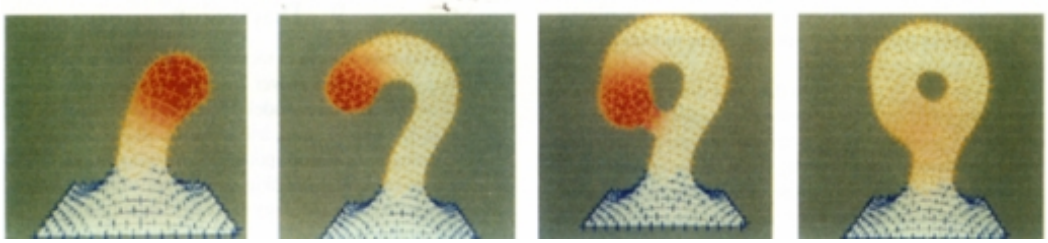


FIG. 1.35: Exemple d'élongation de la surface, puis fusion (figure extraite de [ST92]).

Le maillage maintenu n'est qu'une approximation d'une surface continue, mais c'est lui qui utilisé pour interagir avec la surface sous-jacente, et en évaluer les paramètres :

- Pour maintenir une surface avec de *bonnes* propriétés, on minimise un critère de qualité. Pour chaque sommet du maillage, on reconstruit localement la surface continue :
  - d'abord, le voisinage du sommet est paramétré : pour éviter les instabilités des projections des voisins dans un plan tangent, les auteurs proposent une paramétrisation (r_i, θ_i) où r_i est la distance euclidienne entre le sommet s considéré et son voisin P_i, et θ_i est l'angle en s dans le triangle (P_i, s, P_{i+1}). Les θ_i sont re-normés pour que leur somme soit 2π si s n'est pas un sommet d'un bord (π sinon) (voir figure 1.37.a).
  - puis, la surface est localement reconstruite par un polynôme de degré deux au plus, approchant au mieux les points (au sens des moindres carrés).
  - ensuite, le carré de la courbure principale est minimisé, sans déplacer les points contraints, et la minimisation est itérée au fur et à mesure de l'interaction.
- la triangulation est maintenue par différentes opérations locales simples ne changeant pas la topologie (voit figure 1.37.b) :
  - si le sommet considéré est trop près de ses voisins, il est éliminé (*collapse*). S'il est trop éloigné, un nouveau sommet est inséré (*split*). S'il est plus près de certains de ses voisins seulement, il est replacé géométriquement dans une position *centrale* (évite d'avoir à calculer des forces de répulsions, et de les intégrer dans le temps).
  - si l'échange d'une arête augmente les angles minimums des triangles résultants (au-delà d'un certain seuil, pour éviter les instabilités), cet échange est conservé dans la triangulation (*swap*).

Cette triangulation permet de conserver un échantillonnage uniforme de la surface et évite les triangles singuliers : bénéfice aussi bien pour l'aspect visuel que pour la stabilité des calculs sur la triangulation.

L'utilisateur contrôle la surface en définissant des courbes inscrites sur la surface, ou des régions (des courbes inscrites, fermées).

On peut ensuite joindre deux surfaces en associant deux de leurs courbes inscrites, créer un trou en éliminant tous les éléments du maillage intérieurs à une région.

La définition de l'intersection de deux surfaces ne peut pas être calculée car on ne dispose que d'approximations des surfaces. Cette opération requiert donc d'abord l'intervention de l'utilisateur pour construire une courbe approchant la forme de l'intersection. Ensuite, les deux surfaces sont redéfinies pour l'interpoler.

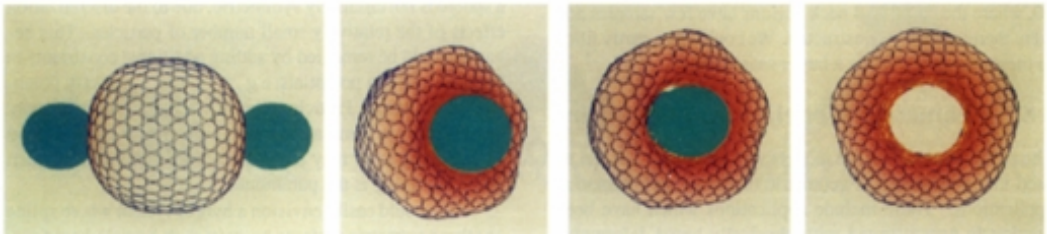
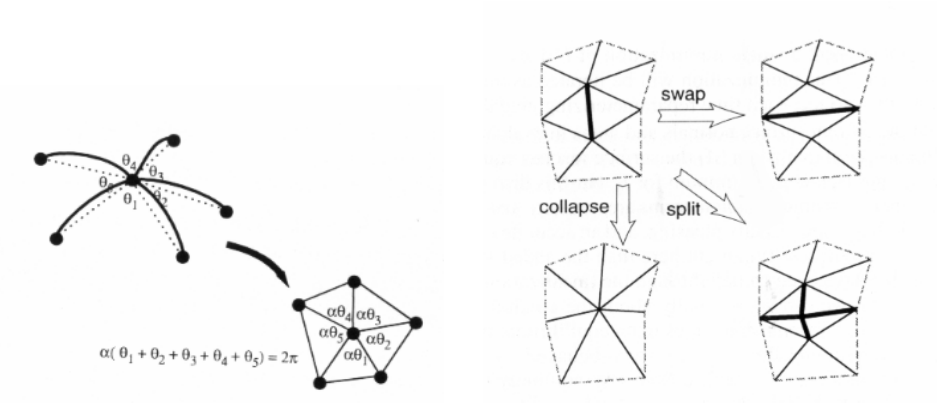


FIG. 1.36: Exemple de suppressions de particules pour changer la sphère en tore avec deux outils (figure extraite de [ST92]).



(a) (b)

FIG. 1.37: (a) : paramétrisation du voisinage d'un point, et (b) : méthodes locales d'évolution de la triangulation sans en modifier la topologie (figure extraite de [WW94]).

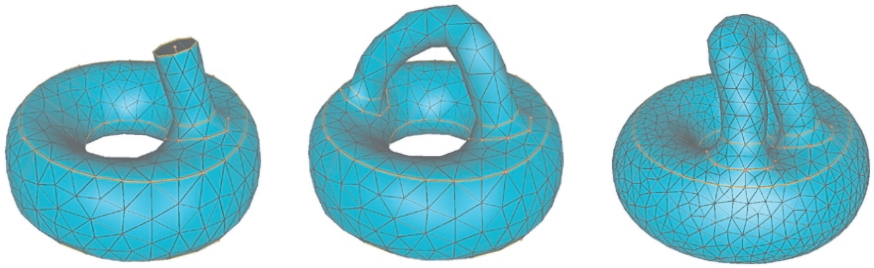
L'article propose aussi de contrôler une région en utilisant une autre surface comme outil. Pour cela, il suffit de contraindre les positions des points intérieurs de la région avec ceux de la surface de l'outil. A l'interface entre les points contraints, et ceux non-contraints de la surface, les conditions de continuité de courbure propageront les forme de l'outil automatiquement.

La formulation ne dépendant que de la triangulation permet également de faire glisser des caractéristiques de la surface (figure 1.38.b et 1.38.c).

A Signal Processing Approach to Fair Surface Design

G. Taubin remarque dans [Tau95] que les méthodes précédentes sont très coûteuses, aussi bien en mémoire qu'en temps de calcul, ce qui les rend inutilisables sur des maillages importants.
Il propose une approche utilisant directement le maillage, sans faire appel à une reconstruction de patches pour évaluer les propriétés de la surface. Il considère de plus le lissage ou *surface fairing* comme une opération de filtrage des hautes fréquences ; dans un contexte de traitement de signal, les éléments de faible rayon de courbure sont en effet décrits par des hautes fréquences.

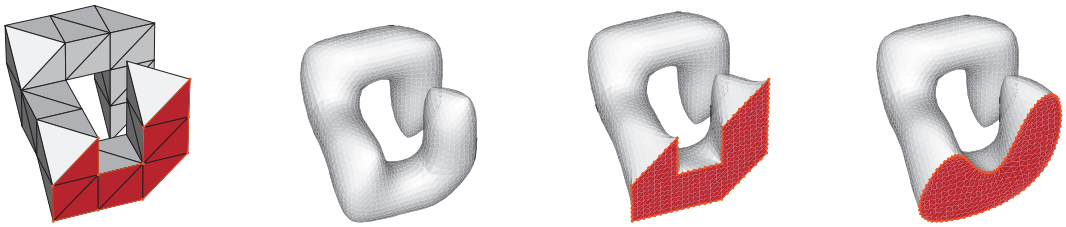
L'article développe des liens théoriques avec l'analyse de Fourier, pour aboutir à un filtre assez simple faisant appel au *Laplacien* discret estimé en chaque sommet du maillage à partir des sommets *voisins* et appliqué itérativement.
La convergence est montrée sur des exemples de maillages assez importants par application d'un nombre d'itérations de l'ordre de la centaine, avec *peu* de changement au delà de 50-100. L'article propose également d'ajouter des étapes de subdivision pour affiner le maillage initial. La méthode proposée se comporte alors comme un outil de construction de forme *lisses* à partir d'un maillage initial grossier. Il réalise aussi par ce biais une analogie avec les surfaces de subdivision (voir section



(a) (b) (c)

FIG. 1.38: Exemple de modification de surface : (a) définition d'un branchement initiant une anse sur un tore, (b) fermeture de l'anse, (c) déplacement d'une des extrémité de l'anse (figure extraite de [WW94]).

1.6) et ramène les schémas de subdivision de Catmull-Clark et Loop à des filtres Gaussiens. Ce développement est assez intéressant car on peut voir la technique exposée ici comme un schéma de subdivision particulier, ou dans l'autre sens les subdivisions *classiques* comme un filtre particulier.



(a) (b) (c) (d)

FIG. 1.39: Exemple de polygone *de contrôle* (a). Les autres figures représentent le même maillage après trois étapes de subdivision / filtrage : (b) sans contrainte, (c) avec des contraintes d'interpolation et (d) avec des contraintes étendues.

L'article développe aussi des astuces pour pouvoir interpoler certains sommets du maillage. Comme le processus de filtrage est obtenu par mises-à-jour successives de la position des sommets en fonction de leurs *voisins*, il suffit d'introduire une relation de *voisinage* non-symétrique. L'idée est de déclarer certains sommets du maillage *sans voisins*. Ainsi leur position ne sera jamais altérée lors du processus de filtrage. Par contre, ces mêmes sommets seront toujours voisins des autres sommets du maillage : ils seront donc pris en compte dans la mise-à-jour de ces autres sommets (voir Figure1.39.b). Notons que les sommets sont toujours voisins en terme de maillage (ils forment toujours une arête), seule leur prise en compte lors du calcul est altérée.
L'extension naturelle de cette astuce consiste à affiner cette idée de voisinage asymétrique en lui associant un ordre. Par exemple, un sommet v_i d'ordre l_i sera déclaré *voisin* d'un sommet v d'ordre l, i.e. utilisé comme tel dans le calcul du filtre, si vv_i est une arête du maillage et $l \leq l_i$. Pour illustrer ce principe, les sommets rouges ont été marqués d'ordre 1, est les autres d'ordre 0 dans la Figure 1.39.d . Ainsi, les sommets rouges sont lissés, mais uniquements entre eux (i.e. ils restent inscrits dans leur plan initial) et la surface les interpole de manière lisse.

A partir de notions de filtrage et d'analyse de maillage, cet article montre donc des potentialités dans la modélisation de surfaces à partir de polygones de contrôle en permettant de préciser des contraintes plus ou moins fortes.
Cette technique trouve des applications et des extensions dans les constructions de hiérarchies à partir de maillages détaillés (comme vu en 1.1). Elle présente également des similarités avec les surfaces de

subdivisions par sa construction. Elle constitue ainsi un lien entre le traitement de signal et l'analyse des surfaces de subdivision, où elle trouve des prolongements comme on le verra en 1.6.

Bilan

Ces techniques cherchant à calculer des surfaces optimales vérifiant certaines contraintes constituent un outil intéressant, car permettant de construire des surfaces *lisses*. A l'exception de la dernière approches, elles présentent toutefois des temps de calculs interdisant une utilisation interactive.
Notons également qu'aucune de ces techniques ne gère les problèmes éventuels d'auto-intersection évoqués précédemment (section 1.3). La surface obtenue n'est donc pas forcément réalisable, comme l'exemple de bouteille de Klein montré dans [WW94].
Enfin, comme évoqué dans la dernière contribution [Tau95], ces techniques sont par essence assez proche des techniques de subivision. On peut ainsi voir la représentation par surfaces de subdivision comme unificatrice, dans un même formalisme, à la fois des surfaces polygonales, des splines et de celles que nous avons appelées surfaces d'optimisation.

1.6 Surfaces de subdivision

Nous proposons de débuter cette courte présentation des surfaces de subdivision avec un exemple illustratif du principe. On verra ainsi le lien avec les surfaces splines évoquées dans la section 1.2.
Nous verrons ensuite à travers un exemple célèbre (le court-métrage d'animation *Geri's Game*) la souplesse de cette représentation dans le cadre d'une utilisation *réelle* de modélisation.
Enfin, nous nous attacherons à montrer les liens avec les modèles polygonaux (section 1.1) via le calcul d'un maillage de base simplifié. Ce maillage simplifié est utilisé comme polygone de contrôle d'une surface de subdivision dont la surface limite réalise l'approximation du modèle polygonal initial. Cette unification des deux représentations splines et polygonales offre un cadre de modélisation multirésolution maintenant bien défini / établi.
Finalement, nous aborderons aussi les liens avec les surfaces d'optimisation et l'interpolation de contraintes abordées à la section 1.5.

Un exemple didactique : schéma de Catmull-Clark

Juste pour donner une idée du principe, suivons un exemple initiatique détaillé dans [HKD93], le schéma de subdivision de Catmull-Clark (voir Figure 1.40). Soit un maillage M^i constitué de sommets

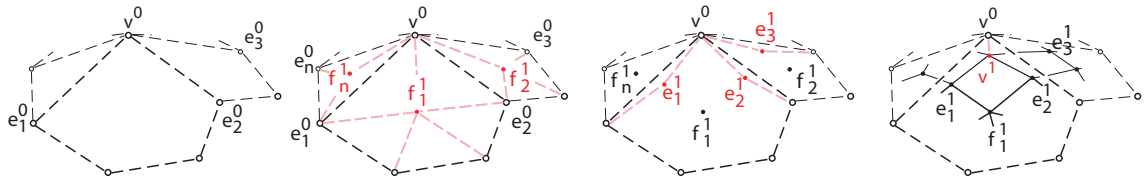


FIG. 1.40: Exemple de schéma de subdivision de Catmull-Clark. Soit un maillage initial M_i, de connectivité et topologie quelconque. On calcule d'abord par interpolation un nouveau sommet par face. On calcule ensuite un nouveau sommet par arête en fonction des sommets du maillage précédent, et des nouveaux sommets par face. Enfin, les nouveaux sommets du maillage sont calculés en fonction de tous ces voisins (figure d'après [HKD93]).

v^j. Pour passer au maillage subdivisé M^{i+1}, on calcule pour chaque face un nouveau sommet f^{i+1} au centre de chaque face. Chaque arête est également découpée par la création d'un nouveau sommet e^{i+1}. Pour une arête j, entourée des faces $j-1$ et j, on a :

$$e^{i+1} = \frac{v^i + e^i_j + f^{i+1}_{j-1} + f^{i+1}_j}{4}$$

Enfin, un nouveau sommet v^{i+1} est calculé :

$$v^{i+1} = \frac{n-2}{n} v^i + \frac{1}{n^2} \sum_j e^i_j + \sum_j f^{i+1}_j$$

Après une application de ces opérations sur un maillage quelconque M^0, le maillage M^1 obtenu est

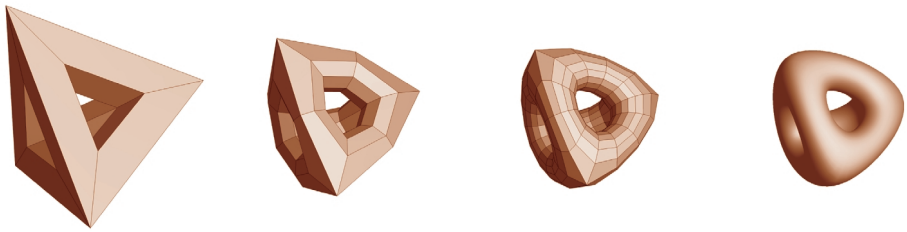


FIG. 1.41: Exemple de subdivision d'un polygone (a) est le maillage de contrôle ; (b) est la première subdivision et (c) la deuxième. (d) est la surface limite (images extraites de [DKT98]).

constitué uniquement de quadrilatères. L'application répétée de ces subdivisions converge vers une surface *lisse* (voir Figure 1.41).

Pour donner une idée de l'utilisation de ces surfaces et de la démonstration de la convergence, il suffit de remarquer que l'application répétée de cette opération ne modifie pas la *valence* d'un sommet (par exemple v^0 dans la figure 1.40). Pour un sommet de valence n, si on utilise $V^i_n = (v^i, e^i_1, ..., e^i_n, f^i_1, ..., f^i_n)$ on peut exprimer V^{i+1}_n comme $V^{i+1}_n = S_n V^i_n$, soit $V^{i+1}_n = S^i_n V^1_n$. A partir de là, il est possible de calculer la position et la normale d'un point de la surface limite $\hat{v}^\infty$ de manière exacte en fonction des points de M_1.

Dans le cas où tous les sommets sont de valence 4, la surface limite est la même que celle obtenue avec des B-splines cubiques. Des équivalences avec les représentations splines selon le schéma de subdivision utilisé sont montrées, par exemple dans [ZSD+00]. L'avantage de ces représentations est qu'elles généralisent les représentation splines en autorisant des topologies quelconques.

J. Stam a montré en 1998 [Sta98] comment étendre l'évaluation exacte des surfaces de subdivision

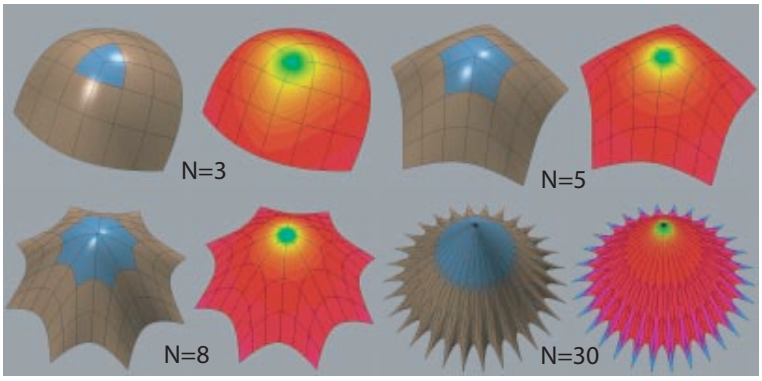


FIG. 1.42: Exemple de surfaces possédant des sommets extraordinaires, i.e. de valence N différente de 4 dans le schéma de subdivision de Catmull-Clark. Les illustrations en fausses couleurs représentent la courbure, également calculée de manière exacte grâce à [Sta98] (images extraites de [Sta98]).

aux sommets qualifiés d'extraordinaires, i.e. de valence différente de 4 dans notre cas, cette valeur

dépendant du schéma de subdivision (voir Figure 1.42).
La prise en compte de ces sommets extraordinaires et des sommets appartenants à des bords (arête ou coin, i.e. jointure de deux arêtes) font appel à des règles particulières, ce qui conduit avec la variété de schémas de subdivision existants à des collections de *recettes*.

Signalons qu'il est possible d'utiliser des faces triangulaires plutôt que *carrées* ou bien encore de diviser les sommets plutôt que les faces (voir Figure 1.43). Il est également possible de varier les

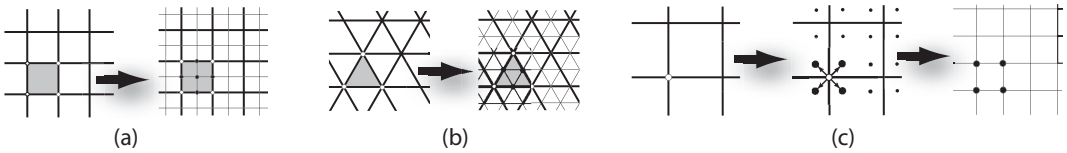


FIG. 1.43: Illustration des principes de subdivision utilisant des faces *carrées* (a), triangulaires (b) ou bien divisant les sommets plutôt que les faces (c) (images extraites de [ZSD+00]).

règles calculant la nouvelle position des sommets à l'issue de la division du maillage. Beaucoup de variations sur ces choix ont été proposées, mais le principe reste le même.

Division par Face		
	Maillage Triangulaire	*Maillage Quad.*
Approximation	Loop (C^2)	Catmull-Clark (C^2)
Interpolation	Butterfly modifié (C^1)	Kobbelt (C^1)

Division par Sommet
Doo-Sabin, Midedge (C^1)
Biquartic (C^2)

FIG. 1.44: Classification des schémas de subdivision proposée par D. Zorin dans [ZSD+00].

D. Zorin propose dans [ZSD+00] une classification parmis les schémas usuellement employés / étudiés (voir Figure 1.44).

Subdivision Surfaces in Character Animation

La popularité des surfaces de subdivisions doit beaucoup au succès de *Geri's Game* [DKT98] qui a montré leur souplesse d'utilisation et apporté quelques contributions pour leur édition / utilisation. T. DeRose et al. exposent dans [DKT98] plusieurs techniques pour notament construire des plis et améliorer leur contrôle, et aussi texturer et animer ces surfaces (détection de collisions, calculs de paramètres physiques sur la surface). Nous allons nous intéresser ici plus particulièrement au premier point concernant les plis.

L'idée dûe à H. Hoppe et al. dans [HDD+94], assez analogue aux astuces de *voisinage asymétrique* de G. Taubin [Tau95] que nous avons vu à la section 1.5, qui est reprise ici [DKT98] et étendue consiste, par exemple pour une arête de *poids n*, à appliquer des règles spéciales pour mettre à jour sa position pendant *n* étapes de subdivision. Pour les subdivisions suivantes, les règles usuelles sont appliquées, ce qui permet facilement de conserver les propriétés de continuité / lissage de la surface limite.
Les règles spéciales n'utilisent que les sommets issus de la subdivision de l'arête considérée dans le calcul de leur nouvelle position. On obtient ainsi par exemple les plis de la Figure 1.45. Un poids ∞ produisant une discontuité, car seule la *règle spéciale* est appliquée.

L'article propose une extension pour utiliser un poids *n* non entier, en faisant une interpolation linéaire entre les positions de la subdivision correspondant à la partie entière de *n* et la subdivision

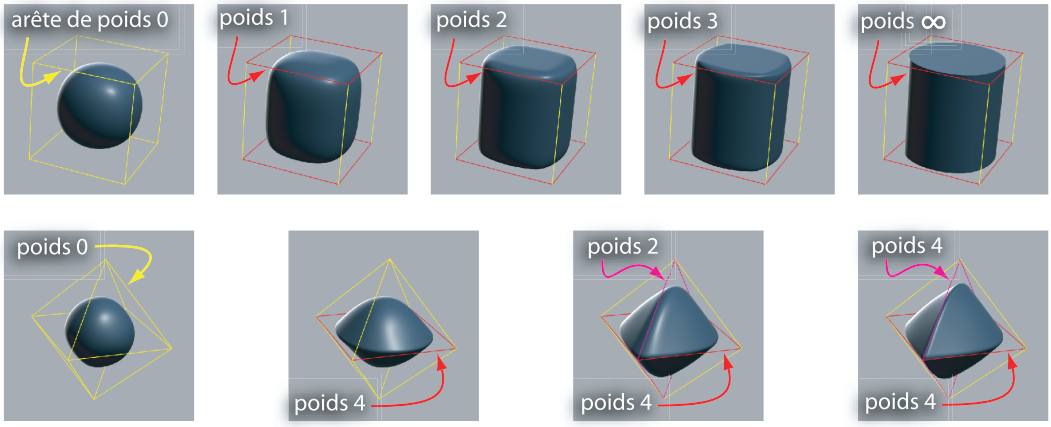


FIG. 1.45: Exemples de contrôle de plis par des pondérations sur les arêtes. Le polygone de contrôle est représenté en jaune et la surface limite obtenue en bleu. Sur la ligne du haut, le polygone de contrôle est un cube dont le poids de certains arêtes varie de 0 à ∞. En bas, exemple sur un octahèdre de pondérations différentes sur des arêtes (images extraites de [DKT98]).

suivante. Ceci est la base pour la seconde extension qui permet de faire varier cette pondération le long d'une arête, utilisé par exemple dans la construction de l'oeil à gauche de la Figure 1.46.

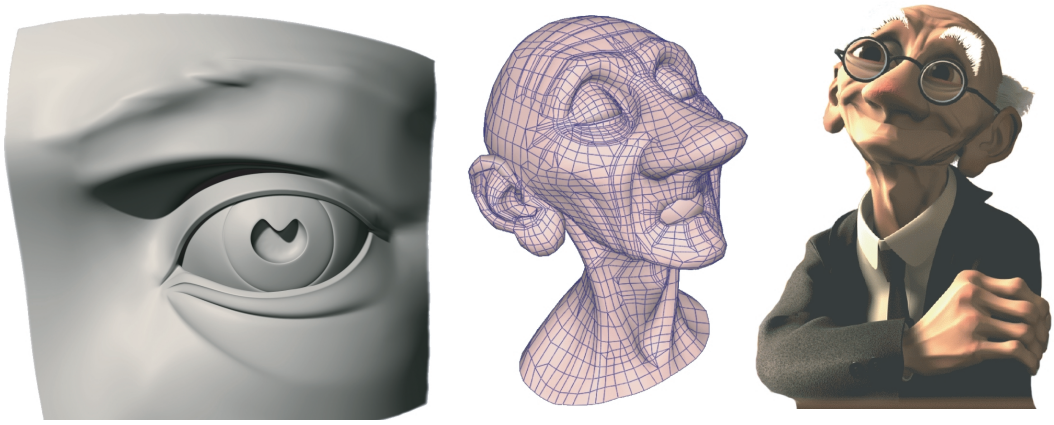


FIG. 1.46: Exemples d'utilisation de surfaces de subdivision avec des plis plus ou moins marqués. A gauche, un modèle utilisant des pondérations variables constitué de 627 faces (contre 840 avec des pondérations constantes). Au mileu et à droite, Geri le personnage du court-métrage de Pixar présenté dans l'article (images extraites de [DKT98]).

Si les approches précédentes révèlent la souplesse et la richesse des surfaces de subdivision dans un contexte de génération de modèles, un problème majeur s'oppose à leur utilisation directe pour éditer / cohabiter avec des maillages quelconques. Si on dispose d'un maillage déjà *détaillé*, i.e. dont les faces sont comparables non pas au polygone de contrôle mais à n étapes de subdivision d'une surface de subdivision, il est impossible de l'éditer *comme* une surface de subdivision. En effet, une surface de subdivision à un niveau de subdivision donné n'est pas équivalente à un modèle polygonal arbitraire, mais présente une connectivité particulière : chaque face est un quadrilatère dans le schéma de Catmull-Clark, ou bien chaque groupe de 4 triangles correspond à un triangle dans la surface précédent cette subdivision. Les deux approches suivantes apportent une solution à ce problème de remaillage qui permet de traiter une surface polygonale quelconque comme une surface de subdivision, c'est-à-dire une hiérarchie de surfaces polygonales.

Multiresolution Analysis of Arbitrary Meshes

La technique de remaillage proposée par M. Eck et al. dans [EDD+95] se déroule en trois étapes :

1. **partition** : le maillage initial M (voir Figure 1.47.a) est d'abord partitionné en régions triangulaires (voir Figures 1.47 b à d), formant un maillage simplifié M^0 (voir Figure 1.47.e).
2. **paramétrisation** : dans chaque région ainsi obtenue, le maillage initial est paramétré localement.
3. **rééchantillonnage** : chaque face du maillage simplifié M^0 est subdivisée, et à chaque étape, les sommets obtenus par subdivision sont corrigés pour interpoler le maillage initial grâce à la paramétrisation obtenue à l'étape précédente.

On obtient ainsi après i subdivisions un maillage M^i proche du maillage M initial, mais possédant une connectivité de subdivision (voir par exemple M^4 Figure 1.47.f). Le partitionnement est réalisé en

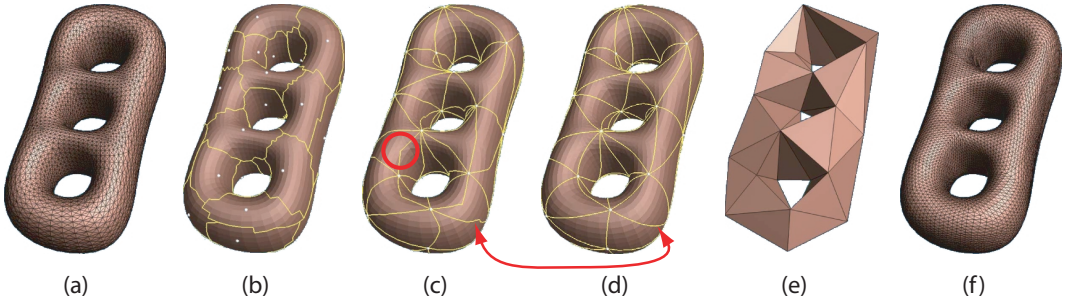


FIG. 1.47: Illustration du processus de remaillage. (a) maillage original (11 776 faces) ; (b) sites trouvés en blanc et pseudo-régions de Voronoï associées en jaune ; (c) pseudo-triangulation de Delaunay approximative ; (d) pseudo-triangulation de Delaunay raffinée (70 triangles) ; (d) maillage de base M^0 ; (e) maillage M^4 (4 subdivisions, 17 920 faces) ; (images extraites de [EDD+95]).

disposant des *sites* sur le maillage. Cette recherche de site se fait de manière *gloutonne* en disposant un site initial (une face) au hasard sur le maillage, qui constitue la liste initiale des sites. Ensuite, chaque site est étendu par insertion des faces voisines formant une région autour du site. La frontière d'une région est constituée d'arêtes du maillage M initial. Cette croissance continue tant que le maillage M n'est pas entièrement couvert, et que les trois critères suivants sont vérifiés pour toutes les régions :

1. chaque région est homéomorphe à un disque. Cette vérification est faite lors de l'insertion de chaque face. Le cas échéant, la face en question devient un nouveau site, i.e. est insérée dans la liste de sites.
2. chaque région a exactement une frontière en commun avec une autre région. Si une frontière est commune à plus de deux régions, un nouveau site est créé le long de cette frontière.
3. pas plus de trois régions ne doivent se rencontrer en un sommet du maillage. Si le cas se présente, un nouveau site est créé à partir d'une des faces adjacentes au sommet.

Une fois cette croissance terminée, on obtient *quelque chose approchant un partitionnement en régions de Voronoï* (voir Figure 1.47). Il est alors possible de construire le dual, constitué des arêtes reliant le centre des régions, qui a alors la bonne propriété d'être constitué de *triangles* (ici, aussi, *quelque chose comme une triangulation de Delaunay*). Ce problème non trivial de recherche de plus court chemin entre deux points sur une surface quelconque est résolu en calculant des cartes $2D$ locales minimisant les distortions (*harmonic maps*) à chaque région. Les demi-arêtes sont alors aisément calulées en $2D$ puis reprojetées sur le maillage M pour chaque région (chemins représentés en jaune dans la Figure 1.47.b). Ces chemins ne sont pas satisfaisants car ils présentent des cassures (voir les annotations en rouge dans les Figures 1.47 c et d). Ils sont corrigés en reconstruisant une carte $2D$ des deux domaines triangulaires adjacents, ce qui donne un carré. Le chemin de l'arête dans cette carte

est remplacé par la diagonale, et re-projetée sur le maillage (Figure 1.47.c). Comme ces chemins ne concordent pas forcément avec des arêtes du maillage M initial, ce dernier est découpé pour contenir les arêtes réalisant le chemin.

On le voit, cette partie de l'algorithme est déjà assez coûteuse. Une fois le maillage de base M^0 obtenu, la subdivision classique permet d'obtenir une surface, qui moyennant des corrections par sommets du maillage M^j pourra approcher le maillage original M. L'approche proposée ici repose sur une analyse par ondelette, issue d'autres travaux des mêmes auteurs [LDW97], que nous ne détaillerons pas ici.

MAPS : Multiresolution Adaptive Parameterization of Surfaces

A. Lee et al. notent dans [LSS+98] que l'algorithme précédent est très complexe et **coûteux**, du fait notament des multiples calculs de cartes $2D$ (*harmonic maps*, qui sont des algorithmes quadratiques), et **fragile** à cause de l'heuristique utilisée pour trouver les *sites* servant au calcul des *pseudo-régions de Voronoï*.
Ils proposent dans cet article une solution utilisant des techniques de simplification de maillage, qui nécessitent également une analyse (paramétrisation) des maillages locales au cours du processus de simplification. Ces techniques sont moins coûteuses que la recherche de *sites* précédente, et permettent surtout de contraindre la décomposition pour suivre des frontières éventuellements précisées par l'utilisateur pour marquer des plis de la surface.

FIG. 1.48: Illustration du principe de l'algorithme de remaillage. (a) maillage M intial ; (b) décomposition M^0 obtenu par simplification ; (c) correspondance entre les sommets du maillage initial M et les triangles du domaine de base M^0 (paramétrisation) ; (d) maillage adaptatif de subdivision obtenu ; (e) opérations d'édition multirésolutions possibles grâce à ce maillage (images extraites de [LSS+98]).

La démarche globale de l'algorithme est essentiellement la même. Un maillage M détaillé de connectivité et topologie quelconque est utilisé en entrée (Figure 1.48.a). Une succession d'opérations de simplification conduit à un maillage de base M^0 (Figure 1.48.b). En même temps que la simplification, une paramétrisation des sommets initiaux locale à chaque triangle de la décomposition est construite (Figure 1.48.c). En utilisant ces paramétrisations, un maillage adaptatif possédant une connectivité de subdivision est construit (Figure 1.48.d). Ce maillage est donc utilisable par les algorithmes d'édition multirésolution de surfaces de subdivision (Figure 1.48.e).

L'article s'attache également à la résolution de problèmes pouvant survenir lors de la mise en correspondance des sommets initiaux dans les simplifications successives (paramétrisation). Ce sont essentiellement des problèmes de retournement de triangles et de distortions dans la subdivision du domaine de base M^0. Les retournements peuvent intervenir car la mise en correspondance effectuée durant la simplification est uniquement calculée pour les sommets du maillage précédent, ce qui par accumulation permet de retrouver la correspondance avec les sommets du maillage initial M, mais la projection d'un triangle de M n'est pas forcément un triangle sur le domaine considéré du niveau j.

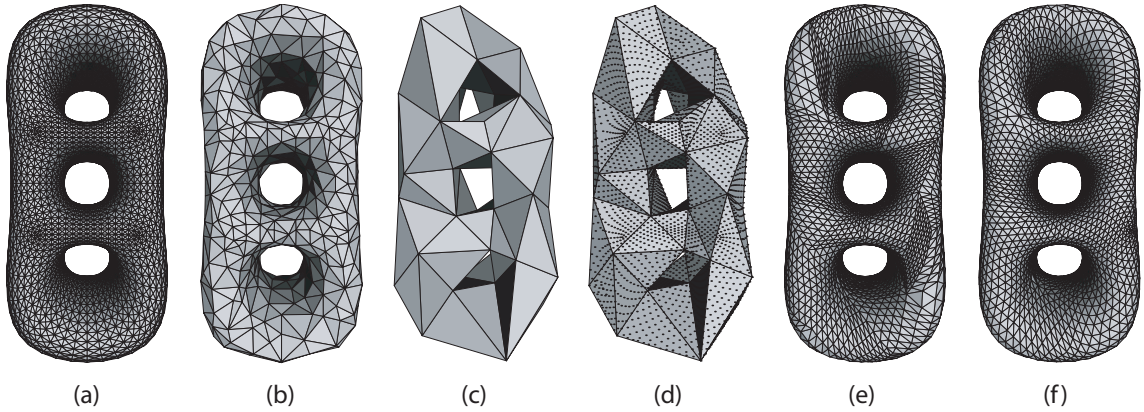


FIG. 1.49: (a) maillage original M, correspondant à un niveau 14 ; (b) niveau intermédiaire (6) de la simplification ; (c) niveau 0 utilisé comme maillage de base M^0 ; (d) M^0 avec les projections des points du maillage initial M sur les triangles de base ; (e) (images extraites de [LSS+98]).

Les distortions quant à elles sont dûes à l'application d'un schéma de subdivision classique (voir Figure 1.49.e). La méthode proposée ici pour minimiser ces distortions est d'adapter les règles de subdivision (calcul de mise-à-jour des sommets) lorsque les sommets voisins utilisés dans de ce calcul traversent un domaine de base. La méthode fait appel à des projections de ces sommets sur une carte locale minimisant les déformations (*géométriques*), ce qui produit un lissage (voir Figure 1.49.f).

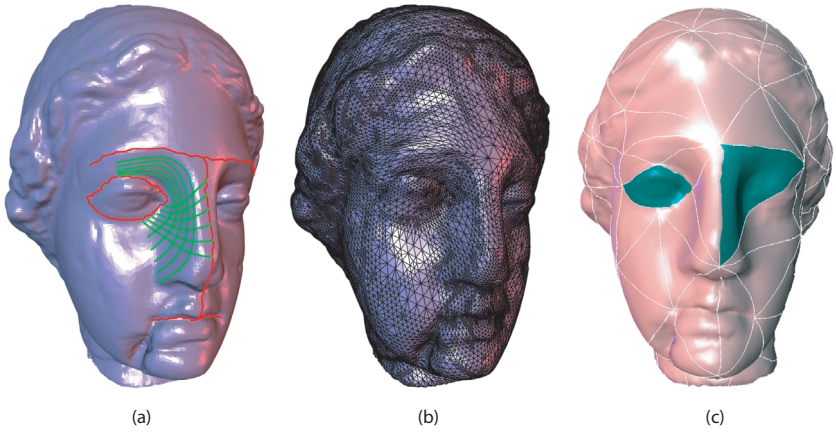


FIG. 1.50: Exemple de remaillage contraint. (a) est la maillage initial (100 000 triangles) avec des lignes dessinées en rouge pour les arêtes contraintes et en vert les lignes iso-paramètres ; (b) est le maillage adapatatif avec connectivité de subdivision obtenu ; (c) montre l'alignement des domaines de base obtenus avec les contraintes (images extraites de [LSS+98]).

Enfin, l'algorithme par simplifications successives présente l'avantage de pouvoir contraindre certains éléments de cette simplification. Par exemple, il est possible de *guider* la décomposition pour conserver des caractéristiques du maillage (Figure 1.50). Ceci est particulièrement intéressant car le contrôle des surfaces de subdivision repose sur des paramètres des arêtes du maillage de base M^0 (par exemple les plis, comme vu précédemment [DKT98]) ; les méthodes de remaillage complètement automatiques ne permettent le placement de telles contraintes, ce qui rend l'édition plus complexe par la suite.

Cette technique déjà intéressante, bénéficie par ailleurs des avancées dans le domaine de la simplification de maillage. Par exemple, les techniques de simplification par des opérations locales d'effon-

drement d'arêtes, guidées par des estimations d'erreur utilisant des quadriques, initiées par M. Garland [GH97, GH98, Hop99] permettent de construire rapidement et efficacement les hiérarchies de maillages exploitées ici.

La contribution suivante illustre les principes et les enjeux de l'édition multirésolution sur des surfaces de subdivision.

Interactive Multiresolution Mesh editing

D. Zorin et al. proposent dans [ZSS97] un système d'édition multirésolution de surfaces polygonales. Ils s'appuient sur une approche de remaillage pour pouvoir travailler sur un maillage polygonal quelconque (phase d'initialisation de la Figure 1.51).

La Figure 1.51 montre un diagramme de fonctionnement du système d'édition proposé. On dis-

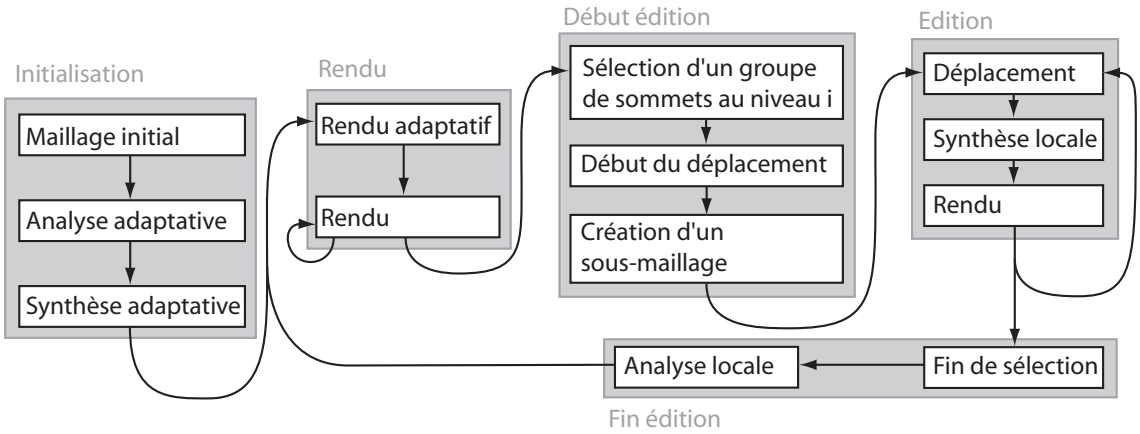


FIG. 1.51: Diagramme de fonctionnement du système d'édition multirésolution interactive proposé par [ZSS97].

tingue une phase d'initialisation, qui consiste à partir d'un maillage quelconque M à extraire la hiérarchie de maillages possédant une conncectivité de subdivision.

L'intérêt de la méthode proposée est qu'elle maintient au cours de l'édition une **colérence** dans la hiérarchie en répercutant les modifications à tous les niveaux. Ceci est réalisé par la phase d'analyse qui conclue une édition. Cette analyse est en fait similaire à l'analyse conduite en initialisation, mais elle est locale à la suite d'une édition (voir Figure 1.52). La représentation utilisée est basée sur des

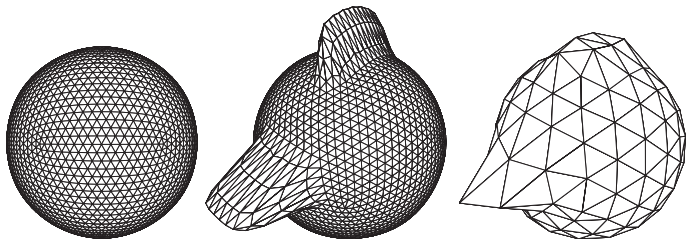


FIG. 1.52: Illustration de la répercution des modifications depuis des résolutions plus détaillées sur les résolutions plus grossières (images extraites de [ZSS97]).

triangles et utilise un schéma de subdivision basé sur des faces triangulaires (Loop).
La répercussion des modifications conduites depuis des échelles plus fines sur les échelles plus grossière (analyse) est réalisée par une technique de filtrage, comme proposée par G. Taubin en 1995

dans [Tau95].
Chaque sommet du maillage contient une information de décalage, *offset* à ajouter à la position donnée par le schéma de subdivision et exprimée dans un repère local au triangle *parent*. Ce repère local est obtenu par des opérations linéaires grâce aux règles de subdivision.

La subdivision n'est pas conduite uniformément sur toute la surface, mais seulement là où des détails existent : l'analyse et la subdivision sont adaptatives. Ces adaptations sont basées sur des seuils. Ainsi, il existe un seuil en deçà duquel les détails sont considérés comme inutiles, c'est-à-dire que la surface obtenue par subdivision seule coïncide *suffisament* avec la surface *originale*. Ceci permet d'éviter de stocker les détails **et** les sommets associés. A l'issue de cette simplification, la hiérarchie n'est pas contrainte : des triangles adjacents peuvent différer de plus d'un niveau de hiérarchie à leur division la plus fine.
Une étape de synthèse est donc nécessaire pour reconstruire les niveaux intermédiaires et assurer qu'un triangle subdivisé jusqu'au niveau i aura pour voisins des triangles subdivisés au moins jusqu'au niveau $i-1$. Ces deux opérations sont conduites localement uniquement à la suite d'une édition.

Finalement, la surface ainsi obtenue est affichée par un rendu également adaptatif. Il est basé sur une évaluation de la distance entre un sommet de niveau i, divisant une arête de niveau $i-1$, et cette même arête pour estimer l'*apport* de ce sommet et choisir de dessiner le triangle de niveau i ou non. L'algorithme gère également les cassures pouvant survenir entre les triangles de niveau de subdivision différents, par l'ajout de *T-vertex* où nécessaire (voir Figure 1.53).

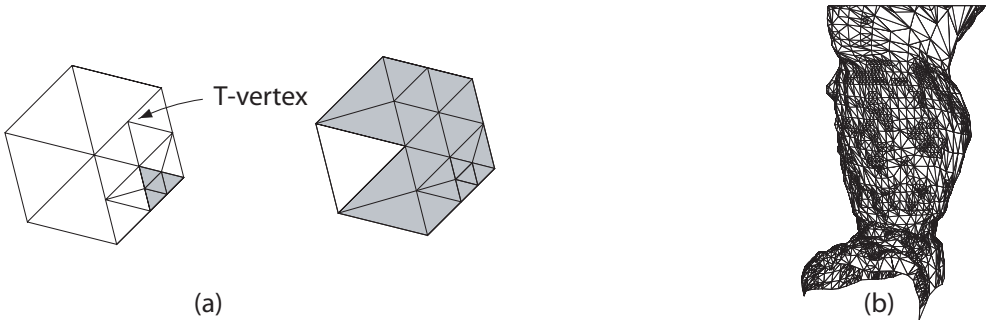


FIG. 1.53: (a). Principe du rendu adaptatif. (b) Exemple de rendu adaptatif (images extraites de [ZSS97]).

Cet article montre la grande souplesse de l'édition multirésolution des surfaces de subdivision, qui permettent d'unifier les représentations splines et polygonales, tout en exploitant efficacement les ressources par des techniques adaptatives.
Nous allons encore aborder deux contributions qui proposent d'autres moyens d'édition faisant appel à des courbes caractéristiques. Ces courbes sont soit spécifiées comme éléments de construction (ce qui réalise une analogie avec les surfaces d'optimisaton vue en 1.5), soit utilisées pour ajouter des détails fins sur une surface existante.

Interpolating Net of Curves by Smooth Subdivision Surfaces

Dans l'approche proposée par A. Levin dans [Lev99], l'utilisateur construit un maillage de contrôle initial pour la surface et un réseau de courbes à interpoler. Certains éléments du maillage doivent être attachés aux courbes (*c-edges* et *c-vertices*). La surface de subdivision est construite grâce à des règles modifiées. Pour calculer les positions des sommets v^j du maillage de $j^{\text{ième}}$ subdivision M^j en fonction du maillage M^{j-1}, des règles particulières sont utilisées si un des sommets pris en compte est attaché à une courbe. Nous ne détaillerons pas ici le catalogue de règles utilisées et nous nous contentons de donner un aperçu (point de vue *utilisateur*) des possibilités offertes par cette approche.

Il est possible de préciser des courbes non continues, ainsi que la continuité de la surface au voisinage des courbes interpolées (via des directions de tangentes dans le plan normal à la courbe). Bien entendu, ces tangentes sont fixées au point où deux courbes s'intersectent par ces mêmes courbes. L'article propose des heuristiques pour les déterminer ces tangentes et éviter à l'utilisateur de les préciser systématiquement.

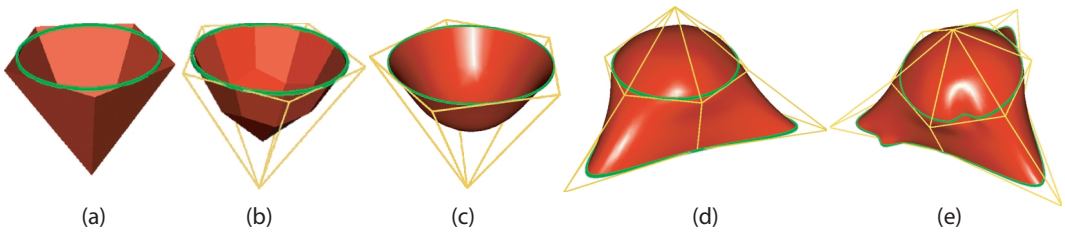


FIG. 1.54: Exemples de surfaces interpolant des réseaux de courbes. Les courbes sont dessinées en vert, et les maillages de contrôle en jaune. (a), (b) et (c) montrent le maillage initial, la première itération et la surface limite obtenue. (d) (images extraites de [Lev99]).

On voit quelques exemples de surfaces obtenues dans la Figure 1.54, et l'intérêt dans le contrôle de la surface en modifiant les courbes interpolées (Figure 1.54 (c) et (d)).
Ces règles de subdivision *modifiées* enrichissent le *catalogue de règles* déjà en place autour des surfaces de subdivisions. Leur analyse fait l'objet d'un autre document que l'on peut trouver dans [ZSD+00] par exemple. La technique proposée ne gère pas l'intersection de plus de deux courbes, mais constitue un outil intéressant pour la construction de formes.

Fine Level Feature Editing for Subdivision Surfaces

A. Khodakovsky et P. Schröder proposent dans [KS99] de compléter les techniques d'éditions offertes par les surfaces de subdivisions en autorisant l'ajout de courbes quelconques sur la surface pour réaliser des extrusion, et ajouter ainsi des détails fins. La définition de ces courbes est complétée par un profil, permettant de contrôler la largeur et continuité des extrusions (voir Figure 1.55).

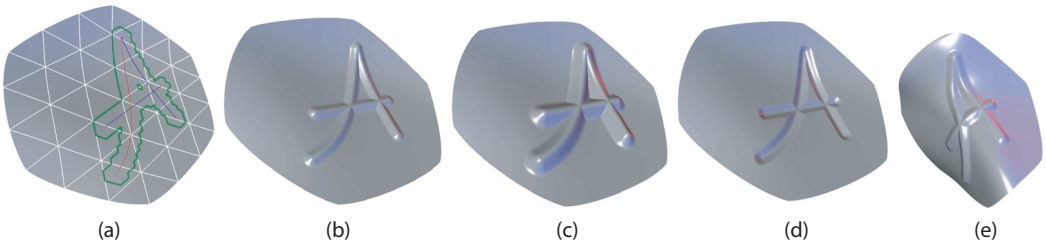


FIG. 1.55: Exemples de courbes dessinées sur une surface de subdivision. (a) montre la résolution éditée avec les trois courbes desinées et en vert le voisinage altéré ; (b) (c) et (d) montrent des variations sur le profile utilisé : positif, plus large, négatif ; (d) les déformations à des échelles plus grossières conservent les courbes définient à des échelles plus fines (images extraites de [KS99]).

Cette approche s'inscrit dans le cadre de surfaces de subdivision enrichies avec des vecteurs de déplacement pour modifier la position des points obtenus par le schéma de subdivision (*Displaced Subdivision Surfaces*).
Dans la démarche proposée, l'utilisateur choisit un niveau d'édition j correspondant à un maillage M^j sur lequel la courbe est dessinée. Un déplacement exprimé dans le repère local à chaque triangle

est calculé et stocké comme déplacement du maillage. Le procédé est répété pour les résolutions inférieures : la subdivision adaptative est effectuée jusqu'à un seuil d'erreur donné entre la surface limite et la triangulation courante.
De telles courbes peuvent co-exister à différents niveaux de la hiérarchie. La définition de courbes dans un niveaux plus fin n'affecte pas les courbes de niveau plus grossier. De même, des modifications sur la surface à un niveau plus grossier que celui où est définie la courbe conservera cette dernière, en l'entraînant avec les modifications (voir Figure 1.56.(d)).

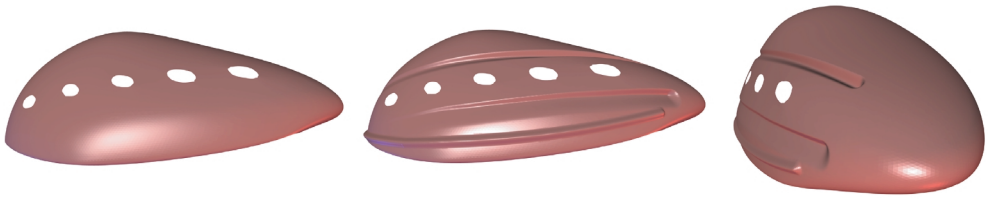


FIG. 1.56: Exemple plus *complexe* d'utilisation de courbes pour réaliser des édition fines. (a) la surface de subdivision initiale. (b) et (c) deux vues de la surface éditée avec des courbes (images extraites de [KS99]).

Il est aussi possible de préciser plusieurs courbes pouvant s'intersecter. Dans ce cas, le déplacement en chaque sommet du maillage M^j est calculé en fonction du maximum (profils positifs) ou du minimum (profils négatifs) des contributions. La Figure 1.56.(a) représente par exemple trois courbes, et en vert le voisnnage impacté par leurs profils. Les auteurs proposent également d'autres variations, notament sur des interpolations de profils le long des courbes. La Figure 1.56 montre un exemple un peu plus complexe de modification à l'aide de ces outils.

Bilan

Ces approches sont très intéressantes, notament du fait des facilités d'édition multirésolution qui n'affectent pas les détails fins déjà construits lors de modifications plus grossières. Elles présentent également des attraits non négligeables dans des domaines voisins, comme la compression ou la transmission progressive [ZSD+00].
Toutefois, la construction et l'édition utilisent toujours un polygone de contrôle qu'il faut manipuler explicitement. On a vu la richesse du contrôle des plissements de la surface, mais il s'effectue justement sur des arêtes de ce polygone. Ainsi, l'utilisateur doit prévoir lors de la construction de la surface la présence dans le polygone de contrôle de ces arêtes aux *bons endroits*. On a vu que des approches spécifiques ont même été développées pour pouvoir préciser de telles arêtes dans le cadre de la construction semie-automatique de ce polygone à partir d'un maillage quelconque.
Une des grandes forces de ces techniques est la topologie libre, mais elle doit tout de même être spécifiée complètement dans le polygone de contrôle. En effet, lors de la subdivision, la topologie ne change pas. Cette contrainte peut amener dans le cas de surfaces complexes à avoir un polygone de contrôle quasiment aussi compliqué que la surface elle-même, et donc difficilement manipulable.

Toutes les approches précédentes permettent des opérations d'édition séduisantes, notament l'analyse multirésolution, la prise en compte de paramèters physiques sous formes de contraintes pour générer des surfaces *optimales*. Toutefois dans toutes les approches que nous avons abordées, le contrôle de la topologie reste à gérer explicitement par l'utilisateur. C'est même souvent **le** maillon faible qui en dépit de toutes les techniques puissantes d'éditions, révèle à l'utilisateur une partie de la représentation. Cette opération nécessite en effet de la part de l'utilisateur un passage par une structure

de contrôle interne à la représentation (elle *démasque* en quelque sorte la représentation sous-jacente). Ainsi, la seule alternative à cette disparition de la représentation qui nous est apparue est l'utilisation des surfaces implicites, que nous allons aborder un peu plus en détails ci-après.

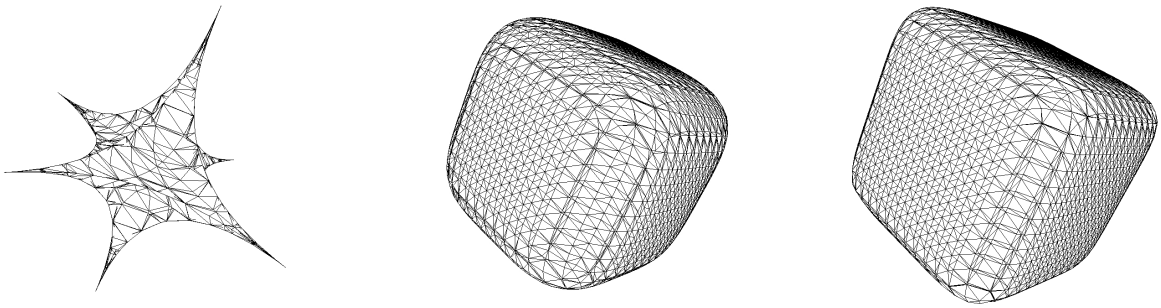
1.7 Modélisation par surfaces implicites

Les surfaces algébriques ont été étudiées depuis plusieurs siècles, mais leur utilisation en synthèse d'images sous l'appellation de *surfaces implicites* ne date que du début des années 1980. Nous allons dans cette partie retracer brièvement cette évolution, pour montrer le cheminement de la modélisation par surfaces implicites et situer le cadre où s'inscrit notre approche.

Définition générale

Une surface implicite est définie à partir d'un *champ scalaire* $f : \mathbb{R}^3 \longrightarrow \mathbb{R}$ comme l'ensemble des points $P(x,y,z)$ tels que $f(P) = c$, où c est une constante appelée *iso-valeur*. Par convention on désigne comme *intérieurs* soit les points P tels que $f(P) > c$, soit les points P tels que $f(P) < c$.

Les surfaces implicites peuvent être définies par une équation algébrique. Cette formulation permet à priori de construire n'importe quelle surface à partir d'un polynôme en x,y,z. Par exemple les super ellipsoïdes d'équation $f(x,y,z) = \frac{x^k}{a^k} + \frac{y^k}{b^k} + \frac{z^k}{c^k}$ ont été utilisées par B. Wyvill et G. Wyvill dans [WW89] (voir figure 1.57). Ou bien, plus récemment, les surfaces de Kummer (figure 1.58) utilisées

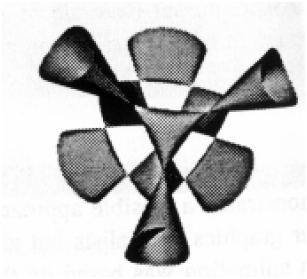


(a) (b) (c)

FIG. 1.57: Exemple de super-ellipsoïdes (a) k=0.5, (b) k=4, (c) k=6, visualisation en *fil de fer* avec faces cachées d'une polygonalisation.

par A. Rösch et al. dans [RRS96].

$$
\begin{array}{rcl}
f(x,y,z) & = & (3-\nu^2)(x^2+y^2+z^2-\nu^2)^2 \\
 & & -(3\nu^2-1)pqrs = 0 \\
\text{avec :} & & p = 1-z-x\sqrt{2} \\
 & & q = 1-z+x\sqrt{2} \\
 & & r = 1+z+y\sqrt{2} \\
 & & s = 1+z-y\sqrt{2}
\end{array}
$$



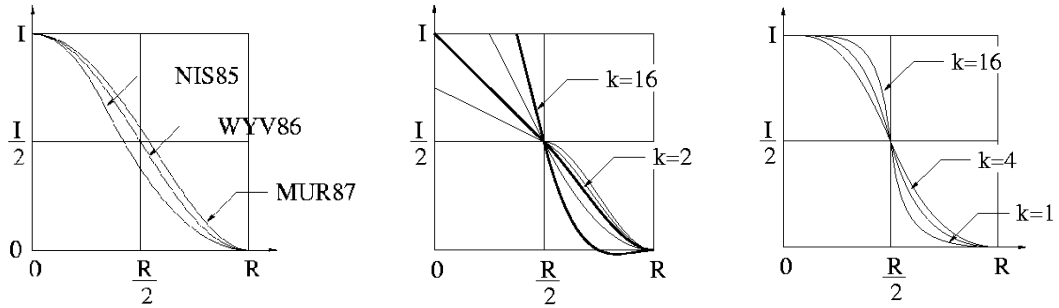
(a) (b)

FIG. 1.58: Surface de Kummer : (a) définition de la famille de surfaces, et (b) exemple pour $\nu^2 = 1.2$, visualisation par lancer de rayons (figure extraite de [RRS96]).

Cette formulation à priori sans limitations, se révèle dans la pratique assez difficilement utilisable, notamment pour le contrôle de la forme : il est délicat de trouver les paramètres mathématiques correspondant à la forme que l'on veut modéliser.

Surfaces implicites à squelette

La première utilisation en synthèse d'images de ce type de surfaces implicites est sans doute dûe à J. Blinn [Bli82] avec les *blobby molecules*. Pour réaliser une animation d'une séquence ADN, J. Blinn propose d'approcher chaque atome par un potentiel gaussien $f(P) = kexp(-\frac{\|P-C_i\|^2}{a})$ où C_i est le centre de l'atome (k et a des constantes), et d'utiliser la superposition des potentiels pour obtenir un champ scalaire, c'est-à-dire la somme : $f(P) = \sum_i f_i(P)$ (source page web de P. Bourke [Bou97]). On obtient ainsi un moyen intuitif de contrôle de la forme à travers les points générateurs et leur *taille* (les coefficient a et k).
Deux publications indépendantes en 1985-1986 marquent une évolution de ce principe : les *Metaballs*, de T. Nishimura et al. [NHK+85] et les *Soft objects* des frères Wyvill [WMW86] (figure 1.59.a).



(a) (b) (c)

FIG. 1.59: Fonctions potentiel : (a) fonctions potentiel proposées par [NHK+85], par [WMW86] et [MI87], (b) par [Can93] et (c) par [BS95] (figures extraites de la thèse de E. Galin [Gal97]).

En fait, les innovations résident dans les fonctions potentiel utilisées pour éviter l'évaluation de l'exponentielle et limiter son extension (contrôle local). Plusieurs solutions ont étés publiées, et plusieurs travaux en font la comparaison (comme [Gal97] ou [BS95]). Toutefois, toutes ces formulations reposent sur le même concept :

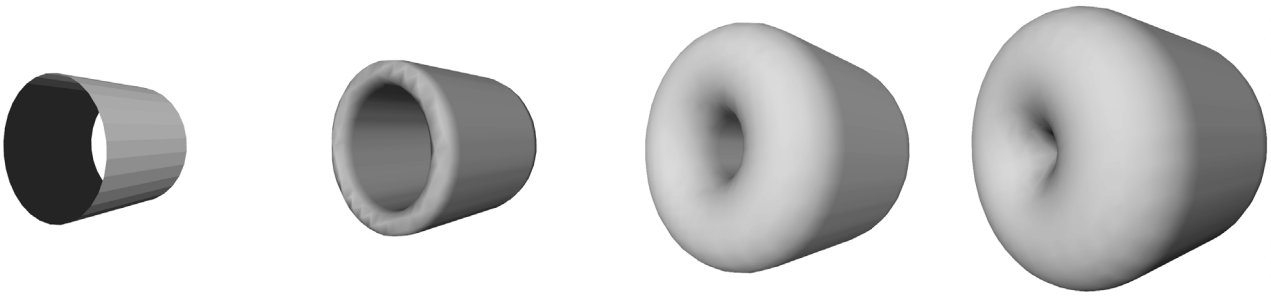
- utiliser un **squelette** S : initialement un point, puis, par extension ([BW90]), un segment, une courbe spline, une surface, et plus généralement tout objet dont on sait évaluer la distance d à un point P donné.
- à partir de cette distance et d'une **fonction potentiel** f, on génère un **champ scalaire** $F_{S,f}(P)$ ayant comme valeur en P $f(d) = f(dist(S,P))$.

L'effet produit correspond à un *enrobage* du squelette (voir figure 1.60).

Mélange

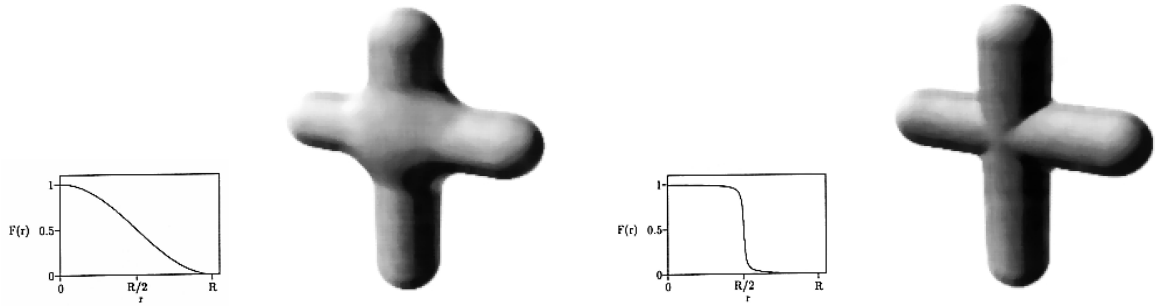
La force et la faiblesse des surfaces implicites réside dans leur aptitude à se mélanger. Plusieurs techniques de combinaison ont été publiées, pour combiner des champs scalaires et donner une surface résultat *intuitive* à partir des surfaces individuelles.
Citons quelques exemples dans le cas de deux champs f_1 et f_2 qui se généralisent aisément à plusieurs champs :

- $\bigcup$, l'union : $f(P) = max(f_1(P), f_2(P))$ qui correspond intuitivement à l'union des deux surfaces S_1 et S_2 associées à f_1 et f_2.



(a) (b) (c) (d)

FIG. 1.60: (a) un cylindre utilisé comme squelette. (b), (c) et (d) les surfaces correspondant à une iso-valeur de plus en plus petite (la surface *s'éloigne* du squelette).



(a) (b) (c) (d)

FIG. 1.61: (a) fonction potentiel *douce*, (b) mélange *somme* correspondant. (c) fonction potentiel *dure*, (d) mélange *somme* correspondant (figures extraites de [KAW91]).

- $\bigcap$, l'intersection : $f(P) = min(f_1(P), f_2(P))$ qui correspond intuitivement à l'intersection des deux surfaces S_1 et S_2.
- $\sum$, la somme : $f(P) = f_1(P) + f_2(P)$, qui donne une surface résultat S joignant S_1 et S_2 de manière C^k (si les champs sont C^k). Z. Kačić-Alesić and B. Wyvill étudient dans [KAW91] l'influence de la forme de fonction potentielle utilisée sur la forme du mélange (voir figure 1.61).
- le jeu de *R*-fonctions évoquées dans [PASS95], et qui unifient les 3 précédentes formulations :

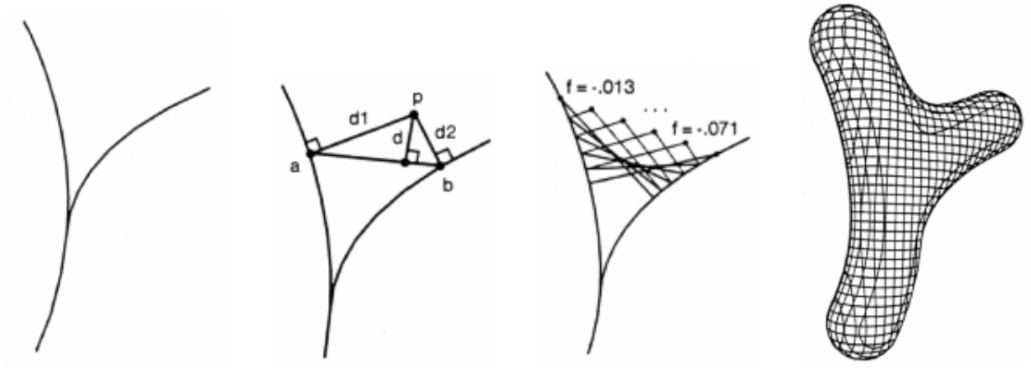
$$f_1 | f_2 = \frac{1}{1+\alpha}\left(f_1 + f_2 + \sqrt{f_1^2 + f_2^2 - 2\alpha f_1 f_2}\right)$$

$$f_1 \& f_2 = \frac{1}{1+\alpha}\left(f_1 + f_2 - \sqrt{f_1^2 + f_2^2 - 2\alpha f_1 f_2}\right)$$

  où l'on retrouve avec $\alpha = 1$, $f_1 | f_2 = max(f_1, f_2)$ et $f_1 \& f_2 = min(f_1, f_2)$.
- les méthodes procédurales, présentées dans [BW90], qui permettent de définir des *distances composées* d'un point au squelette. Par exemple dans le cas de deux segments, on peut définir la distance d'un point P au squelette formé des deux segments comme la distance au segment défini par les deux projetés de P sur le squelette (voir figure 1.62).
- la méthode de convolution, introduite par J. Bloomenthal dans [BS91] pour palier au comportement *non superposable* de la somme : $f_{S_1} + f_{S_2} \neq f_{S_1+S_2}$ (voir figure 1.63).
  Pour un squelette S, le champ scalaire en P se calcule par :

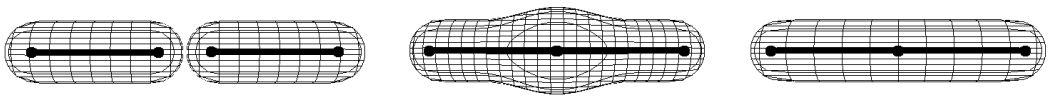
$$f_S(P) = \int_S exp\left(-\frac{\|S-P\|^2}{2}\right).ds = (h * S)(P)$$

  qui est la convolution de S par $h(M) = exp\left(-\frac{\|M\|^2}{2}\right)$. Ainsi, on a bien la propriété de superpo-



(a) (b) (c) (d)

FIG. 1.62: Mélange procédural : (a) le squelette composé de deux courbes. (b) calcul d'une distance *procédurale* : la distance d'un point P au squelette est évalué comme la distance au segment formé par les deux projetés P_1 et P_2 de P sur chaque segment. (c) illustration de la variation de cette distance quand P se déplace le long d'une droite. (c) iso-surface obtenue avec cette distance composée (figures extraites de [BW90]).



(a) (b) (c)

FIG. 1.63: Non superposition de la somme de potentiels : (a) deux iso-surfaces des champs scalaire générés par deux segments S_1 et S_2. Lorsqu'on joint les deux segments, la surface obtenue pour la somme des champs f_1 et f_2 associés aux demi-segments S_1 et S_2, en (b), est différente de la surface que l'on obtiendrait pour le champ généré par le squelette S union de S_1 et S_2, en (c).

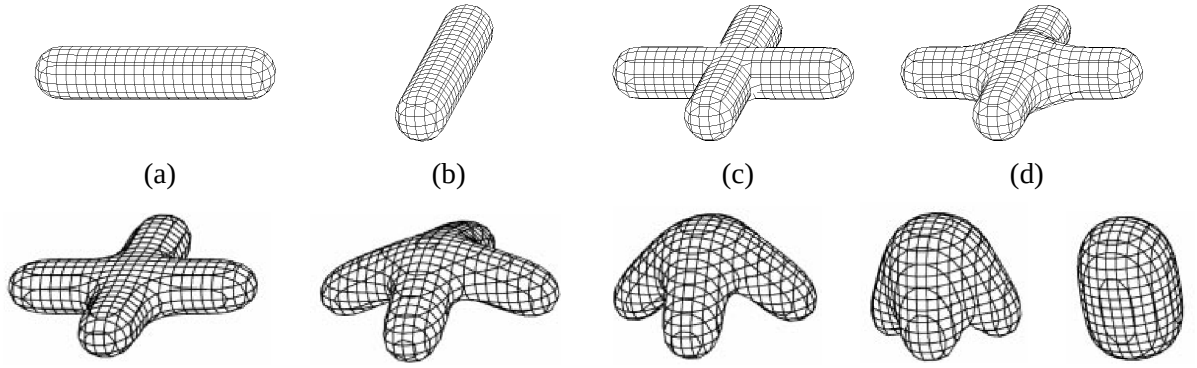
sition recherchée (voir figure 1.64) :

$$f_{S_1}(P) + f_{S_2}(P) \leftrightarrow (h * S_1)(P) + (h * S_2)(P) = (h * (S_1 + S_2))(P) \leftrightarrow f_{S_1+S_2}(P)$$

Affichage

J. Bloomenthal et B. Wyvill évoquent également dans [BW90] les procédés d'affichage de des surfaces implicites.

1. Le plus lent, mais sans doute celui fournissant la meilleure qualité est le lancer de rayons. D. Kalra and A. H. Barr proposent dans [KB89] d'utiliser les propriétés du champ scalaire pour assurer de ne pas manquer d'intersections d'un rayons avec la surface. Grâce aux constantes de Lipschitz :
   – L du champ scalaire f dans une région rectangulaire de l'espace, c'est-à-dire $\|f(x_1) - f(x_2)\| < L\|x_1 - x_2\|$.
   – G de la dérivée directionnelle du champ le long d'un rayon.
   l'article montre comment subdiviser efficacement l'espace et calculer l'intersection des rayons avec la *surface-LG*.
2. J. Bloomenthal et B. Wyvill signalent dans [BW90] ou [Blo95] l'existence de visualisation par *iso-contours* de la surface implicite ([Ric73]) qui consiste à découper l'espace par une série de plans parallèles, chacun contenant un iso-contour intersection avec la surface.
3. le rendu volumique, issu des domaines médical et de simulation de l'éclairage (simulation de milieux participants) peuvent bénéficier d'accélérations matérielles spécifiques comme (source [Mah97]) :



(a) (b) (c) (d)

(e) (f) (g) (h) (i)

FIG. 1.64: (a) et (b) deux champs scalaires générés par deux segments, et une iso-surface correspondante. (c) la surface union. (d) la surface somme, on remarque la bosse centrale. (e) la surface convolution correspondante, avec de (f) à (i), les repliement des demis-segments.

- les texture-3D de Silicon Graphics et apparues récemment sur les cartes vidéos standards, exploitant les accélérations des textures, basées sur l'utilisation de plans de textures traversant les données volumiques parallèlement au plan de projection. Plusieurs rapports et articles comme [CN93] et [WvGW94], et plus récemment [WE98] abordent l'utilisation de ce moyen de rendu de données volumiques.
- le *Voxelator* de Hewlett-Packard, basé sur les techniques de *ray-casting*, classiques en rendu volumique.
- la carte *Cube4* de l'université d'état de New-York au *Center for Visual Computing* de SUNY/Stony Brook, sujet de la thèse doctorale de H. Pfister sous la direction de A. Kaufman, qui possède une grande mémoire de données volumiques, et *sait* en extraire une projection [PK96] et son évolution VolumePro [PHK+99].

4. polygonalisation, une des représentation qui permet d'exploiter les accélérations graphiques des stations. Quelques étapes marquantes :
   - J. Bloomenthal dans [Blo87], ou W.E. Lorensen and H.E. Cline dans [LC87] présentent l'algorithme dit de *marching cubes* qui consiste, à partir d'un cube initial intersectant la surface, à suivre cette surface par exploration des cubes voisins. Chaque cube est traité indépendamment pour trianguler la portion de surface qu'il contient.
   - J. Bloomenthal expose dans [Blo87] et [Blo88] un procédé de subdivision adaptatif utilisant comme critère le nombre de polygone par cube, la planarité de ces polygones ou bien encore l'écart entre les normales aux sommets et la normale des facettes trouvées, pour guider la subdivision des cubes en cubes plus petits.
   - P. Ning and J. Bloomenthal étendent dans [NB93] le procédé en partitionnant les cubes en six tétrahèdres pour réduire le nombre de cas à traiter : ils passent de 256 configurations pour trianguler un cube à 16 pour un tétrahèdre, mais les triangles générés sont de moins bonne *qualité* (angles minimum plus petits) et leur nombre est plus important.
     J. Bloomenthal reprend plus en détail cette méthode dans [Blo94] et en donne une implémentation (également disponible sur *l'implicit site* : [BH98]).

5. les méthode par *particules*, évoquées pour la première fois dans [BW90] permettent également d'exploiter les accélérations graphiques des stations et connaissent un renouveau depuis que le maintien interactif d'une triangulation est envisageable.
   - A. Witkin and P. Heckbert utilisent dans [WH94] des particules *flottantes* qui suivent la surface, et des particules de contrôle, qui permettent d'affecter les paramètres du champ pour que l'iso-surface suive la particule de contrôle déplacée par l'utilisateur.
     Les particules flottantes n'ont pas de relations de voisinage fixées : elles sont contraintes sur

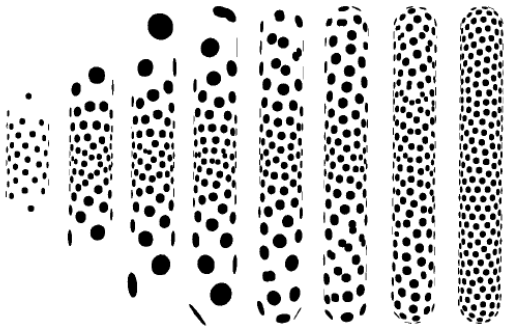
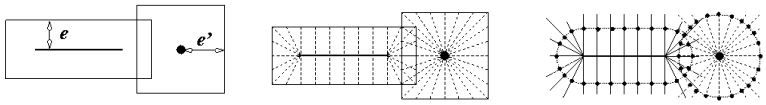


FIG. 1.65: Exemple d'un cylindre s'allongeant rapidement, les particules flottantes suivent son déplacement et rétablissent un échantillonnage régulier de la surface (figure extraite de [WH94])

la surface et se repoussent entre elles avec une énergie $E_{ij} = \alpha exp\left(-\frac{\|r_{ij}\|^2}{2\sigma^2}\right)$ où α est un paramètre global d'amplitude du terme de répulsion, et σ est le rayon de répulsion.
A chaque pas de temps, la particule se déplace dans la direction qui minimise cette énergie, conduisant itérativement à une solution minimale. σ permet de régler la densité d'échantillonnage (les distances entre les particules) et ainsi ouvre la voie à un échantillonnage adaptatif en temps (raffinement par division des particules de faible vitesse, donc proches d'un équilibre), et en espace (selon des critères de courbure, par exemple). Fixer σ donne un échantillonnage régulier de la surface (voir figure 1.65).

– Dans le cas de surfaces implicites à squelette avec des fonctions potentiel possédant un rayon d'influence (c'est-à-dire qui sont nulles au-delà de ce rayon), il est possible de déterminer une *boîte d'influence* en dehors de laquelle le champ scalaire de chaque squelette est nul (voir figure 1.66.a). M. Desbrun et al. proposent dans [DTC96] d'utiliser cette boîte englo-

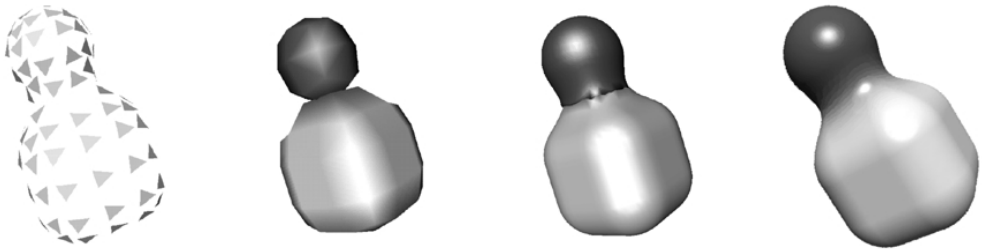


(a) (b) (c)

FIG. 1.66: M. Desbrun et al. utilisent dans [DTC96] la *boîte d'influence* (a), c'est-à-dire la boîte englobante du squelette augmentée du *rayon d'influence* de la fonction potentiel (*e* dans la figure (a)). Cette *boîte d'influence* permet d'initialiser les positions des particules, leur *direction de propagation* et aussi la triangulation (les relations entre les particules) (b). Ensuite, les particules convergent vers la surface par recherche dichotomique le long d'un axe (c).

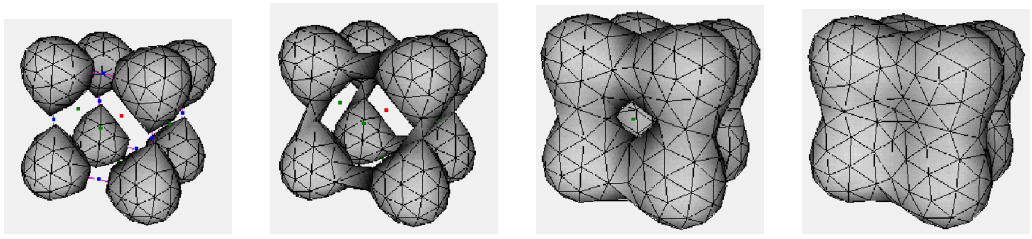
bante pour fixer la position initiale des particules, leur direction de migration (le gradient du champ, i.e. vers le point du squelette le plus proche du point considéré) et les relations entre les particules (une triangulation) restant fixes pour des squelettes simples (voir figure 1.66). Ces particules *migrent* ensuite selon leur axe soit vers l'iso-surface, soit vers la surface délimitant la zone de prédominance de l'élément simple de squelette référence, offrant ainsi une triangulation par morceaux de la surface implicite (voir figure 1.67).

– B.T. Stander and J.C. Hart reprennent dans [SH97] le principe exposé dans [WH94] et maintiennent une triangulation sur les particules flottantes, calculée dans une phase initiale. Pour maintenir cette triangulation, ils proposent d'identifier, puis de suivre au cours de la phase interactive les points critiques du champ scalaire. Les points critiques sont identifiés grâce à la *matrice de stabilité* (la hessienne du champ scalaire). Le signe des valeurs propres de cette matrice donne le *type* du point critique : point *maximum*, *selle-2*, *selle-1* ou *minimum*. L'article montre comment utiliser ces points pour détecter les changements de topologie de



(a) (b) (c) (d)

FIG. 1.67: Résultats obtenus par la méthode de [DTC96] : (a) visualisation des particules par des facettes triangulaires tangentes à la surface, (b) et (c) visualisation de la triangulation obtenue avec plus ou moins de particules, (d) visualisation de la même surface par lancer de rayons.



(a) (b) (c) (d)

FIG. 1.68: Gestion de la correction de la topologie de la triangulation d'une surface générée par huit points squelette : quand un point *selle-1* est traversé par la surface (i.e. la valeur du champ dépasse l'iso-valeur) la surface se connecte en ce point (de (a) à (b)). Quand un point *selle-2* est traversé par la surface, un trou est comblé (de (c) à (d)) (figures extraites de [SH97]).

la triangulation, et les gérer correctement (voir figure 1.68).

Contrôle de la forme des surfaces implicites à squelette

L'apport principal des surfaces implicites à squelette est la facilité d'édition qu'elles procurent. Ainsi, il est possible de contrôler la forme de l'iso-surface générée par :

- déplacement des squelettes sous-jacents : la surface est alors mise à jour en conséquence et les changements de topologie comme les fractures et les fusions sont automatiquement pris en compte.
- modification de la forme d'une ou plusieurs des fonctions potentiel utilisées (figure 1.61).

De manière plus générale, la forme d'une surface implicite peut être modifiée en :

- appliquant une transformation de l'espace $T : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ comme décrite dans [Bar84] à son champ scalaire F : cela revient à composer F avec la transformation inverse de T, le champ transformé en P s'évaluant par $F(T^{-1}(P))$.
- ajoutant un terme de déformation local au champ scalaire. Cette dernière technique a été employée pour modéliser des collisions, comme nous allons le voir ci-après.

Plusieurs environnements on été proposés pour intégrer ces facilités d'édition et de modélisation sous forme d'arbre (voir Figure 1.69).

Modélisation de collisions

M.-P. Cani s'intéresse dans [Can93] à la déformation de deux *solides implicites* élastiques S_i et S_j (réciproquement $f_i = c$ et $f_j = c$) lors d'une collision, et propose de simuler cette collision comme suit :

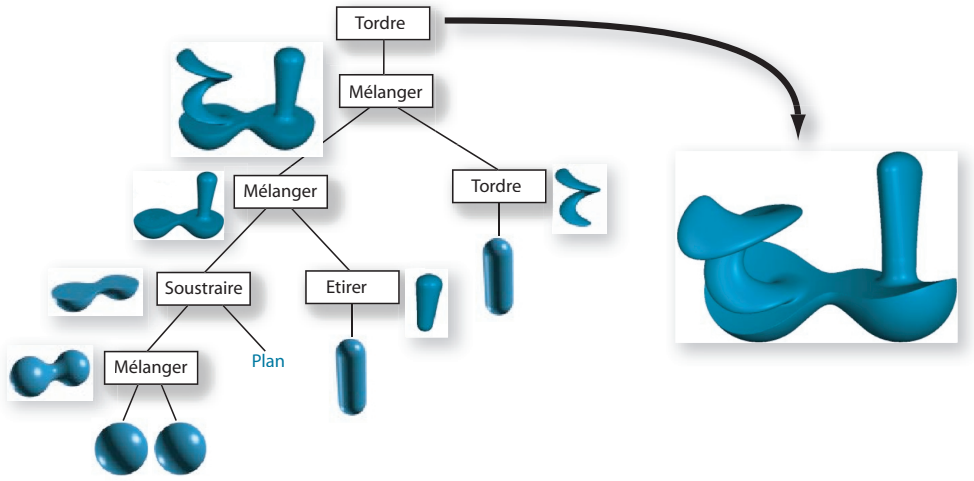
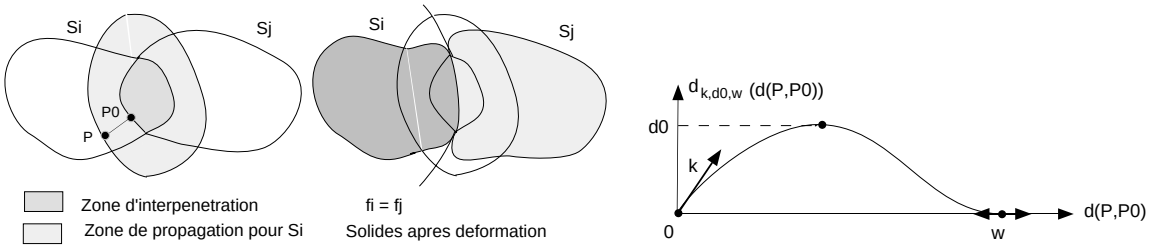


FIG. 1.69: Exemple de décomposition d'un objet implicite (image extraite de [WGG99]).



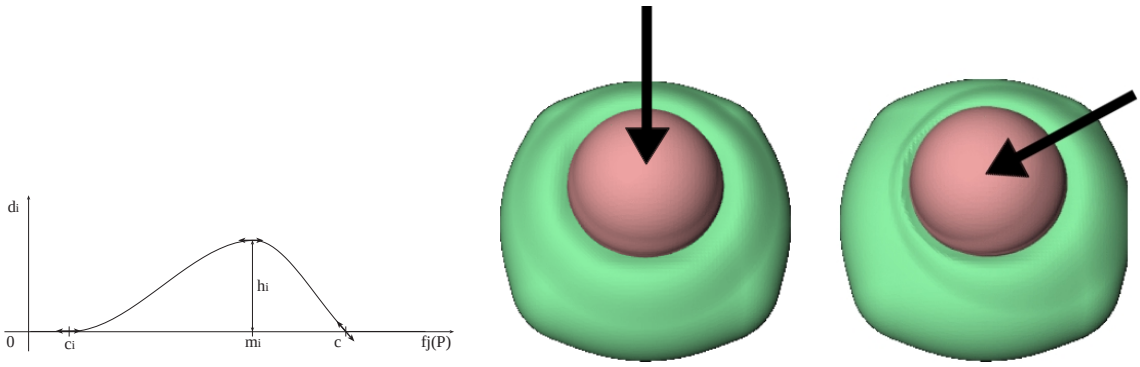
(a) (b)

FIG. 1.70: (a) définition des zones d'interpénétration et de propagation (à gauche), et modélisation de la surface de contact et des dilatations (à droite). (b) fonction utilisée pour simuler les dilatations (figures extraites de [Can93]).

- ajouter des potentiels de compression $(c - f_j)$ à f_i et $(c - f_i)$ à f_j, donnant comme expression de la surface de contact $f_i = f_j$ (voir figure 1.70.a, à droite)
- les dilatations au bord de la surface de contact sont traitées géométriquement, par exemple pour S_i, en ajoutant un terme à f_i de la forme $d_{k,d_0,w}(d(P,P_0))$ (voir figure 1.70.b) où $d(P,P_0)$ est la distance de P à P_0, le point *projeté* de P sur la zone d'interpénétration dans la direction du gradient de f_j (voir figure 1.70.a). En fait P_0 est obtenu par itérations, c'est-à-dire par une succession de déplacements dans la direction du gradient de f_j jusqu'à atteindre l'iso-valeur de la surface S_j.
  L'utilisateur contrôle d_0, la déformation maximale, et w, l'extension de cette déformation.
  Le paramètre k est fixé par les conditions de continuité du raccord avec la surface de contact : le gradient du terme de dilatation doit être égal au gradient du terme de compression en P_0, ce qui conduit à $k = \|\nabla f_j(P_0)\|$.
- les forces de réaction sont calculées le long de la surface de contact, la pente de la fonction potentiel autour de l'iso-valeur c étant utilisée pour coder l'élasticité du matériau modélisé : $R_i = -R_j = -(f_j - c).N_i$

A. Opalach and M.-P. Cani modifient dans [OC97] le terme de dilatation précédent dans un double objectif :

- simplifier son calcul : Au lieu de calculer le point P_0 le *plus proche* de la zone d'interpénétration, on utilise directement $f_j(P)$ comme mesure de l'*éloignement* à la surface S_j. On remplace ainsi la recherche itérative du point P_0 et ses nombreuses évaluations du potentiel par une seule évaluation. Le terme de dilatation ajouté à f_i devient alors $d_i(f_j(P))$ (voir figure 1.71.a), dont on peut contrôler h_i, la déformation maximale, m_i la position de cette déformation maximale



(a) (b) (c)

FIG. 1.71: (a) fonction utilisée pour simuler les dilatations ; exemples de déformation simulée : (b) avec une collision *frontale* d'une sphère sur un bloc déformable et (c) avec une collision où la sphère vient de droite (figure extraites de [OC97]).

par rapport aux bords, et c_i l'extension de la déformation.
La continuité C^1 au raccord avec la surface de contact (terme de compression) est assurée par la pente -1 de d_i en P_0 tel que $f_j(P_0) = c$ qui donne bien le gradient du terme de dilatation $\nabla(d_i(f_j(P_0))) = d_i'(f_j(P_0)).\nabla f_j(P_0) = d_i'(c).\nabla f_j(P_0) = -\nabla f_j(P_0)$ égal au gradient du terme de compression.

– modéliser des **directivités** : en utilisant le paramètre c_i contrôlant l'extension de la dilatation de S_i, et la vitesse relative D_{ji} de S_j par rapport à S_i.

$$c_i(P) = \frac{c_0}{2}\left(1 + \frac{D_{ji}.\nabla f_i(P)}{\|D_{ji}\|.\|\nabla f_i(P)\|}\right)$$

On voit sur la figure 1.71 les types de déformations obtenue.

Les deux approches précédentes modélisent des contacts de manière géométrique entre des surfaces implicites élastiques, ce qui semble séduisant pour simuler l'interaction d'un outil avec un bloc déformable ; la sculpture est même évoquée comme perspective de [OC97].

1.8 Sculpture avec des surfaces implicites

Après ce bref aperçu des définitions et techniques de modélisation par surfaces implicites, nous allons nous concentrer sur les approches exploitant ces techniques dans un cadre de sculpture vituelle interactive.

Utilisation de primitives à squelette

S. Mizuno et al. présentent dans [MOiT98] un système de sculpture utilisant des primitives CSG définies par des plans ou des ellipsoïdes pour obtenir un objet implicite initial. Les outils quant à eux sont des ellipsoïdes qui peuvent être soustraits ou ajoutés à la forme de base.
Ce type d'approche génère une nouvelle primitive implicite dans la représentation de l'objet sculpté à chaque opération. Comme l'affichage requiert l'évaluation du champ potentiel scalaire, cette évaluation deviendra de plus en plus complexe au fur et à mesure de l'édition.
Pour pallier cette augmentation du coût d'évaluation, les auteurs exploitent leur technique d'affichage. En effet, les objets implicites sont affichés par lancé de rayons simplifié : chaque pixel de l'écran correspond à un rayon dans l'espace objet. Sur chacun de ces rayons une liste de points d'intersection avec la surface est maintenue. Les opérations d'édition se réduisent donc au recalcul local des pixels

modifiés en mettant à jour la liste des intersections associées.

Ceci permet de maintenir une vue (ou plusieurs, les auteurs utilisent deux vues par exemple) interactivement, mais présente le désavantage d'interdire les déplacements interactifs de la caméra. En effet, un déplacement de caméra forcerait au recalcul de tous les pixels, et donc de leurs rayons associés dans l'espace objet. Cette opération devant forcément repasser par une évaluation complète de toutes les primitives (car on ne possède plus la cohérence des listes de points d'intersection) sera donc non interactive, et ralentie avec le nombre de primitives (opérations d'éditions).
L'article détaille ensuite des techniques d'encrage virtuel en déposant une feuille de papier sur les sculptures réalisées, qui sont hors de notre intérêt ici.

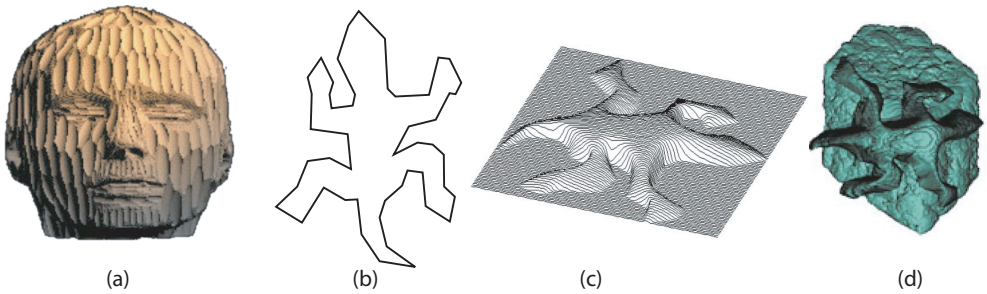


FIG. 1.72: Illustration des formes obtenues dans [PSS01]. (a) est une tête *sculptée* par soustractions successives d'ellispoïdes. (b) est un polygone utilisé pour construire une carte de profondeur (c), qui à son tour peut être utilisée pour modifier une forme (d) (images extraites de [PSS01]).

Des techniques assez proches en ce qui concerne la modélisation sont proposées dans [PSS01]. A. Pasko et al. étendent les outils disponibles. Ils proposent en effet d'utiliser des modèles polygonaux, des cartes de profondeur, des courbes $2D$ et proposent de les combiner avec le modèle en utilisant le jeu des *R-fonctions* (voir Figure 1.72). A. Sourin dans [Sou01] reprend ces principes en adaptant les primitives et *R-fonctions* pour reproduire l'effet des outils utilisés dans la sculpture de bas-reliefs. Dans ces deux approches, comme les objets sont représentés avec l'intégralité des modifications sous forme de *R-fonctions*, l'évaluation du potentiel devient très coûteuse avec le nombre d'opérations réalisées. Les auteurs exploitent ici aussi l'affichage par lancé de rayons pour tirer profit de la localité des modifications dans l'espace écran et ne redessiner que la partie altérée. Ceci couplé avec des techniques de boîtes englobantes pour limiter les calculs permet une mise à jour interactive de l'affichage (ici aussi, dans la limite où la vue reste constante).

Ces contributions sont à notre connaissance les seules utilisant des primitives implicites à squelette et/ou enrichissant le modèle initial de chacune des modifications.
Les auteurs exploitent implicitement le fait que les modifications sont uniformément réparties, et en faible nombre dans une zone donnée. Dans le cas contraire, la mise-à-jour demandant une réévaluation même locale risquerait de devenir trop coûteuse pour maintenir l'interactivité.
Nous avons préféré des approches cumulatives, où la complexité de l'évaluation du modèle stocké reste constante quelque soit le nombre d'opérations. Un moyen pour parvenir à cette fin est d'utiliser des surfaces implicites discrètes. Cette solution a déjà été explorée, comme nous allons le voir dans les contributions suivantes.

Sculpting : An Interactive Volumetric Modeling Technique

T.A. Galyean and J.F. Hughes proposent en 1991 dans [GH91], un *environnement* de sculpture virtuelle reposant sur une formulation implicite, sans s'intéresser à la simulation de contacts entre les

outils et l'objet sculpté.

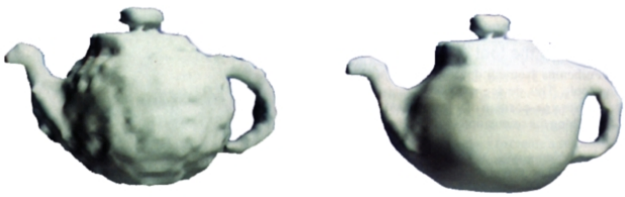
La surface implicite n'est pas définie par des squelettes, mais par un champ scalaire discret échantillonné aux nœuds d'une grille régulière. La surface est ensuite affichée grâce à un algorithme de type *marching cubes*, conduit sur la même grille régulière. Ainsi, il n'est pas nécessaire d'échantillonner le champ potentiel, et la structure servant à le stocker est utilisée également pour afficher la surface.

Dans cette approche, modifier la surface est équivalent à modifier les valeurs du champ stockées aux nœuds de la grille. Les outils 3D modifient ces valeurs comme un pinceau modifie les pixels dans un logiciel de dessin 2D. Cette analogie avec les logiciels de dessin 2D (*bitmap*) pour modifier l'objet (*voxmap* dont les valeurs varient entre 1.0, présence de matière, et 0.0, pas de matière) permet à l'utilisateur familier des pinceaux 2D de comprendre facilement le fonctionnement des outils proposés.

Les outils sont de forme cubique ou sphérique, et ont plusieurs actions possibles :
- effacer : outil *gomme* qui enlève de la matière en annulant les valeurs du champ aux sommets des voxels concernés.
- ajouter de la matière : outil *tube de dentifrice* qui permet de déposer de la matière le long de la trajectoire de l'outil.
- fondre la matière : l'outil *pistolet à chaleur* (*heatgun*), qui creuse l'objet progressivement en diminuant les valeurs des sommets de la grille qu'il recouvre.
- lisser : outil *toile émeri* qui permet de lisser la partie de l'objet incluse dans l'outil en appliquant un filtre paramétrable (voir figure 1.73).

L'application de l'outil dans la grille de voxels pose des problèmes de filtrage (*aliasing*) que les auteurs résolvent par des techniques classiques de sur-échantillonnage de l'outil (quatre fois la résolution de l'objet) et de filtrage (deux passes d'un filtre 2^3). Cette approche se révèle toutefois problématique dans le cas d'outils directionnels, qui nécessiteraient de re-échantillonner et re-filtrer l'outil à chaque changement d'orientation, ce qui interdit ce type d'outil ici.

Typiquement, le système propose un mode basse résolution où le champ scalaire est défini par un tableau de $10 \times 10 \times 10$ voxels, et un mode haute résolution de $30 \times 30 \times 30$ voxels. L'interactivité est possible car les modifications sont **locales** à l'espace délimité par l'outil. Ceci permet d'accélérer l'algorithme de marching cubes : comme on connaît la position de l'outil, on sait quels *voxels* ont été modifiés. Il est alors possible de ne traiter **que** ces voxels pour mettre à jour la discrétisation de la surface, ce qui économise les opérations coûteuses de suivi de surface (*marching*) et réduit aussi le nombre de *cubes* à traiter. Les auteurs baptisent cet algorithme **incremental marching cubes** : un seul marching dans une phase initiale, puis une succession de mise à jour locales.



(a) (b)

FIG. 1.73: Exemple de *sculpture* obtenue dans [GH91]. (a) une théière 9244 polygones, (b) la même après lissage avec la *toile émeri*.

L'article exploite encore la localité des modifications en subdivisant l'écran en régions pour ne re-dessiner que les régions invalidées par une mise à jour de la surface. Quand le point de vue change,

toutes les régions sont invalidées. Les auteurs signalent que si le système permet d'obtenir rapidement une forme quelconque sans limitations de topologie, il ne permet pas une grande précision et n'est pas adapté à la modélisation de pièces mécaniques par exemple (voir figure 1.73).

Parmi les extensions envisagées dans la conclusion de l'article, citons :
- l'implémentation d'opérations de *haut niveau* comme la torsion, les étirements, les opérations de copie, la gestion d'une pile de défaire/refaire,...
- la gestion de la résolution : l'idée des auteurs est d'arriver à une grille de résolution telle que la projection d'un voxel est inférieure à un pixel, pour éviter le coût d'un rendu par polygones.
- la gestion d'une hiérarchie de grilles pour ne raffiner que là où cela est nécessaire.

Volume Sculpting

S.W. Wang et A.E. Kaufman reprennent des principes similaires en stockant les objets sur une grille, et enrichissent le jeu d'outils disponibles dans [WK95].

Les outils sont toujours représentés par des objets binaires (valeur 1 à l'intérieur et 0 à l'extérieur) pré-calculés et préfiltrés. Les outils sont typiquement échantillonnés sur des grilles 20^3 et la taille est ajustée non pas en ré-échantillonnant l'outil sur une grille plus étendue, mais en déformant la grille initiale. L'orientation et la taille des outils est variable. L'application d'un outil consiste ici aussi à combiner les valeurs interpolées de l'outil avec celles de l'objet pour mettre à jour ces dernières.

Le système offre aussi la possibilité de dessiner des courbes d'extrusion sur l'écran. La trace de la région $2D$ dessinée est projetée et échantillonnée dans l'espace objet. Ici aussi, cet échantillonnage est binaire (1 dans la zone intérieure à la région extrudée, 0 en dehors). L'article détaille donc des techniques de filtrage (convolutions exploitant des tables de précalcul) pour éviter les artefacts de crénelage lors de l'extrusion.

FIG. 1.74: Exemples d'image obtenue dans le système proposé par [WK95]. (a) représente quelques étapes de construction d'une chaise *sculptée* dans un block de bois de résolution $75 \times 125 \times 75$. (b) représente un scène constituée de deux objets et (c) une scène un peu plus complexes rendue en ray-tracing (ombres et réflexions) (images extraites de [WK95]).

Le système gère également plusieurs objets, stockés comme autant de volumes de position et orientation arbitraires, mais ne décrit pas comment sont prises en compte leurs interactions éventuelles (intersections ou mélanges).

Contrairement à l'approche précédente, le rendu est ici effectué par *ray-casting* ce qui permet un affichage interactif en cours d'édition, mais pas le déplacement de la caméra. La gestion du filtrage pour les pixels des bords des objets utilise les valeurs filtrées stockées dans le volume pour faire varier

l'opacité. Cela produit des arêtes floues (voir Figure 1.74) et ressemble aux rendus volumiques classiques utilisés pour afficher des opacités (données médicales par exemple), ce qui n'est pas vraiment adapté pour le rendu de surfaces.

A Haptic Interaction Method for Volume Visualization

R.S. Avila et L.M. Sobierajski présentent en 1996 dans [AS96] un système très similaire. L'outil et l'objet sont représentés par des volumes de valeurs scalaires. Ils introduisent un retour d'effort, sur lequel nous reviendrons dans la partie 2.4. Les voxels contiennent donc une densité et un attribut de couleur, comme dans les approches précédentes. Pour réduire l'utilisation mémoire, le gradient du champ potentiel scalaire est stocké via une table de correspondances pour la direction et une valeur pour la norme. Ils ajoutent aussi une propriété du matériau utilisée pour coder une dureté du matériau. Ils proposent plusieurs *outils*, cette dénomination regroupant ici à la fois la forme et l'action. Ainsi,

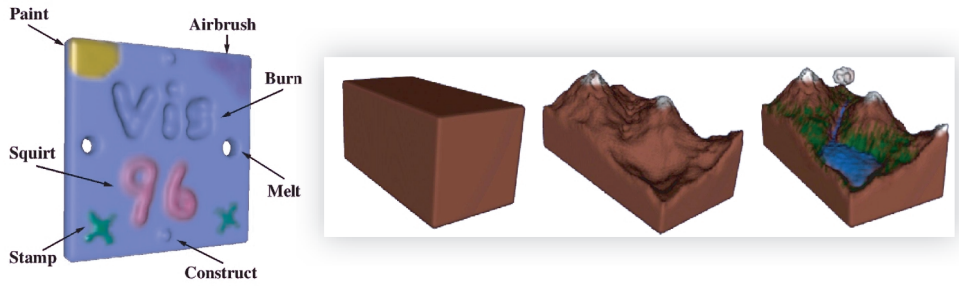


FIG. 1.75: Exemples (a) d'outils proposés dans [AS96] et (b) trois étapes de *sculpture* d'une scène de montagne (images extraites de [AS96]).

en combinant des actions sur le potentiel scalaire (ajouter ou enlever de la matière), sur les attributs de couleurs (combiner les couleurs ou les remplacer, altérer les propriétés comme la brillance) et la forme (une sphère ou une croix) ils proposent septs outils différents (voir Figure 1.75.(a)).

Les trois approches précédentes sont très intéressantes, car la complexité d'évaluation du potentiel ne varie pas avec le nombre d'opérations conduites sur le modèle. Les travaux qui suivent exploitent en plus la multirésolution pour pouvoir affiner les opérations.

Three Dimensional Freeform Sculpting via Zero Sets of Scalar Trivariate Functions

A. Raviv et G. Elber dans [RE99, RE00] proposent une représentation sensiblement différente. Le potentiel scalaire est représenté de manière continue par des patches B-Spline triparamétriques réguliers et indéformables. Ces patches sont de taille et orientation/position arbitraires. Les contribution des différents patches se superposant sont sommées. La surface sculptée correspond aux zéros du potentiel résultat.

Le volume de travail est fixé initialement et subdivisé par un octree. Cet octree est utilisé à la fois pour l'affichage par un algorithme de marching cube dans ses cellules, et aussi pour localiser efficacement les patches dans une zone de l'espace (référence aux patches concernés dans les cellules).

L'édition se fait à l'aide d'outils quelconques représentables par une équation implicite pour évaluer rapidement les points à l'intérieur. L'édition se faisant un patch à la fois, il est nécessaire de sélectionner un *patch actif* avant cette opération. Ensuite, les coefficients du patch actif sont modifiés.

Comme l'outil est représenté uniquement par ses valeurs *dedans/dehors* (i.e. de manière binaire), il est ici aussi nécessaire de filtrer sa représentation discrétisée dans le repère du patch actif pour mettre à jour correctement les coefficients.

Une fois cette modification effectuée, la surface est mise à jour grâce à l'octree, qui doit échantillonner les patches B-Splines en ses sommets. Ceci implique une évaluation coûteuse faisant appel à des multiplications de fonctions de base. Les auteurs choisissent de répercuter ce coût en mémoire en stockant ces multiplications. Comme à priori les patches ne coïncident pas avec l'octree, les points d'échantillonnage sont susceptibles de varier d'un patch à l'autre, ce qui peut devenir extrêmement coûteux. Les auteurs s'intéressent également à des modifications de l'algorithme de Marching Cube utilisé par cellulle pour recoller les triangles non concordants entre deux cellules adjacentes de résolution différentes.

Cette approche présente à notre sens plusieurs points discutables. Tout d'abord, la double représentation par patches et octree n'apporte pas vraiment de points intéressants : le potentiel scalaire est bien représenté de manière continue, mais comme les opérations de sculpture sont uniquement conduites sur les coefficients des patches, il n'est par exemple pas possible de faire *sortir* la surface de ce patch. Egalement, la sélection avant chaque édition d'un patch actif à modifier, exhibe la représentation, ce que nous souhaitons ici éviter. L'approche proposée perd de plus un des attraits des techniques précédentes en séparant le stockage du champ potentiel de l'échantillonnage en octree pour son affichage.

Virtual Clay : A Real-time Sculpting System with Haptic Toolkits

K.T. McDonnell et al. dans [MQ00, MQW01, MQ02] développent une approche également basée sur des volumes B-Splines triparamétriques, mais ils exploitent cette représentation à la fois pour générer la surface correspondante, par subdivision, et conduire une simulation physique du matériau décrit.

Nous ne détaillons pas les principes décrits dans les articles de simulations par masses-ressorts ou d'intégration explicite/implicite. Nous nous intéressons plutôt aux techniques d'éditions qui en découlent. Ainsi, il est possible de déformer la surface en accrochant une *corde incompressible* à un des *points de masse* de la surface et de l'utiliser comme un ressort pour déformer la surface *correspondante*. Des manipulations de ces points de plus haut niveau peuvent être obtenues en les attachant à une courbe spline dessinée par l'utilisateur ; l'édition de cette courbe permettant de déplacer avec elle les points attachés.

Il est aussi possible de paramétrer la simulation physique en *peignant* sur la surface des raideurs ou masses variables. Cette simulation peut être désactivée localement cellule par cellule ou en positionnant une *fenêtre cubique* définissant la portion du modèle où est menée la simulation. Concernant les opérations plus géométriques, elles se font explicitement à partir d'éléments du polygone de contrôle. La Figure 1.76 montre des exemples d'extrusion (ajout de cellule) et de destruction de cellule. La destruction de cellules est libre, ce qui permet de modifier la topologie de l'objet modelé. La Figure 1.77.(b) montre un exemple de changement de topologie par connexion de deux faces sélectionnées du polygone de contrôle. La Figure 1.77.(a) montre un exemple de contrôle de la continuité de la surface de subdivision associée par modification d'un attribut d'une face.

Les problèmes pouvant survenir avec ce type d'approche sont du même ordre que ceux liés aux surfaces de subdivision. Ils proviennent de l'interaction explicite avec le polygone de contrôle, comme on l'a vu beaucoup d'opérations recourent à la sélection d'une face ou d'une cellule. Cela peut poser problème si par exemple l'utilisateur n'a pas prévu dès la construction la présence ou l'alignement

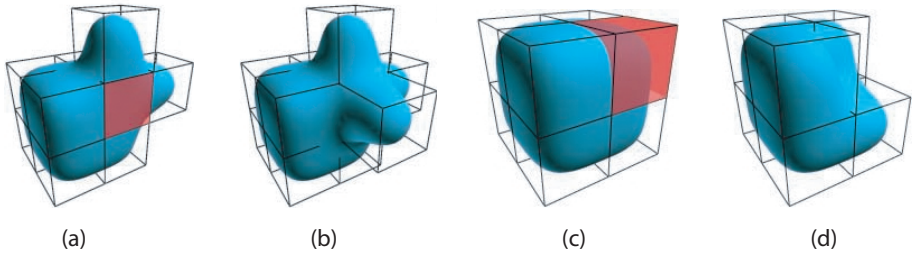


FIG. 1.76: Exemple de modifications apportées à la surface. (a) sélection d'une face, et (b) extrusion de la surface selon cette face. (c) séléction d'une cellule et (d) destruction de cette cellule (images extraites de [MQ02]).

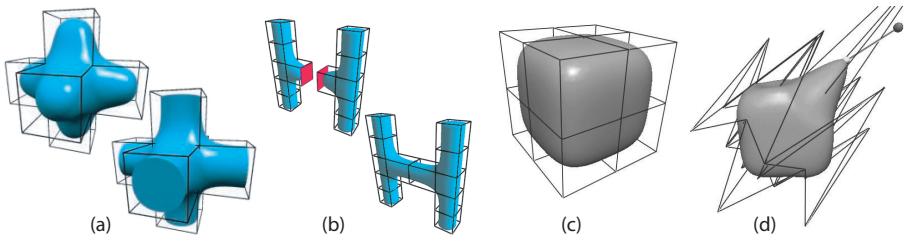


FIG. 1.77: (a) contrôle de la continuité de la surface en marquant certaines faces comme saillantes et (b) modification de la topologie par connection de deux faces sélectionnées du polygone de contrôle (images extraites de [MQ02]). (c) et (d) exemple de déformations *physiques* pouvant entraîner d'importantes déformations du polygone de contrôle associé (images extraites de [MQ00])

du polygone avec les caractéristiques souhaitées. Supposons par exemple dans la Figure 1.76.(b) que l'on souhaite réaliser l'extrusion un peu plus à gauche, mais pas au bords (i.e. la cellule voisine). On pourrait artificiellement s'en sortir en subdivisant les cellules, mais cela compliquerait inutilement le modèle. La simple sélection de faces ou cellules peut également devenir compliquée si le modèle est complexe, ou bien suite à une déformation, comme l'illustre la Figure 1.77.(d).

Après cette parenthèse sur l'utilisation de volumes B-Splines triparamétriques, revenons aux représentations discrètes, qui présentent à notre sens l'avantage de ne pas laisser apparaître de structure de contrôle.

Octree-based Volume Sculpting

J.A. Bærentzen présente dans [Bær98] une approche multirésolution *plus économe* que [RE99] car basée sur une représentation par octree uniquement. Le volume de travail est ici aussi fixé initialement et la résolution de l'octree est au maximum de 256^3. L'affichage est assuré par une technique de ray-casting, qui permet un affichage interactif des modifications mais pas de mouvements de caméra, comme on l'a déjà signalé. Le lancé de rayon est ici accéléré en utilisant la structure d'octree qui permet d'éviter les régions vides sans descendre jusqu'au voxel *feuille* de la hiérarchie.

Les outils utilisés ici sont représentés par des fonctions continues au lieu de discrétisations binaires initiales, puis filtrées comme dans les approches précédentes. La mise-à-jour n'exploite pas l'octree pour mettre à jour des niveaux grossiers en premier, et l'outil travaille à chaque opération sur les voxels de résolution la plus fine. Le regroupement de voxels de valeurs similaires n'est également évoqué que comme extension future.

Cette contribution est intéressante, mais elle ne tire pas pleinement profit de la hiérachie maintenue.

Kizamu : A System For Sculpting Digital Characters

R.N. Perry et S.F. Frisken dans [PF01a] reprennent la représentation qu'ils avaient proposée l'année précédente dans [FPRJ00] pour l'étendre et l'utiliser dans un cadre de sculpture interactive. La représentation en question repose sur des informations volumiques discrètes stockées dans une structure hiérarchique. Au lieu de stocker un potentiel scalaire comme pour les surfaces implicites, c'est la distance euclidienne signée à la surface qui est calculée et stockée.

L'article [PF01a] présente quelques optimisations par rapport à [FPRJ00] pour le calcul de cette distance. Notament, le premier article divisait complètement l'espace pour calculer la distance et procédait ensuite par simplification selon un seuil d'erreur pour simplifier la représentation et aboutir à une hiérarchie adaptative de type octree. La solution proposée ici procède par *briques* (*tiles*) qui sont une division arbitraire du volume englobant initial. Chaque brique est ensuite divisée si elle contient un élément de surface, ou bien selon un critère d'erreur. Le critère d'erreur consite à comparer la différence entre la distance reconstruite (par interpolation) et la distance calculée à un seuil d'erreur. La division est arêtée par un autre seuil de profondeur maximale de la hiérarchie.

L'édition est possible soit directement à partir des triangles de la surface, soit par des opérations CSG sur le champs de distance. Pour assurer l'interactivité lors de ces opérations, le champ de distance est mis à jour uniquement au voisinage de la surface. La mise à jour globale est assurée dans les moments d'inactivité du système.
L'affichage est assuré soit par une triangulation de la surface, soit par des techniques d'affichage par points. Les problèmes de déchirures pouvant apparaître lors de la triangulation de cellules adjacentes de résolution différentes sont également abordés.
L'article propose aussi des méthodes d'édition *contraintes* telles que le suivi de surface, ou la contrainte en position de l'outil à une distance d de la surface. Egalement, la possibilité de constuire des *chemins* d'outil comme des courbes de Bézier, ou bien par des techniques de scripts.

Contrairement aux approches précédentes [GH91, WK95, AS96], la représentation par distance signée utilisée ici présente quelques désavantages. En effet, le potentiel scalaire des surfaces implicites est la composition de la distance avec une fonction de support fini R et qui décroît pour s'annuler de manière continue avec la distance en R. Ceci présente l'avantage de limiter l'étendue des modifications lors d'un changement de la surface. Dans l'approche proposée ici, un changement même minime de la surface peut causer un changement non borné de la distance euclidienne. Egalement, dans le cas où la distance présente des discontinuité, le critère d'erreur peut ne pas être satisfait et entraîner la subdivision jusqu'au fond de la hiérarchie inutilement. De telles discontinuités dans le champ de distance peuvent même se produire arbitrairement loin de la surface, entraînant cette division dans des zones inintéressantes. Les auteurs contournent ce dernier problème en limitant la mise à jour avec un seuil sur la distance à la surface dans la mise à jour interactive.

Bilan sur la modélisation

On a vu dans cette partie la richesse et la diversité des représentations existantes et des techniques d'édition pour la modélisation de surfaces. La plupart des représentations abordées permettent plus ou moins facilement une édition directe de la surface produite, essentielle dans notre contexte de sculpture virtuelle.

Les techniques de FFD offrent une grande souplesse pour effectuer des modifications globales sur un objet existant. Elles montrent par contre leurs limites lorsqu'on souhaite éditer plus finement un modèle plus complexe, comme on l'a évoqué.
Nous avons vu que les surfaces de subdivision peuvent être vues comme l'unification des représentations polygonales et splines. Elles offrent des moyens puissants de contrôle de la surface et d'édition multirésolution. Toutefois, ces opérations font appel à des paramètres précisés sur le polygone de contrôle. Comme on l'a évoqué, ce polygone peut avoir une topologie arbitraire, mais elle doit être la même que celle de l'objet final ce qui peut conduire à un polygone très complexe.
La principale limitation dans notre contexte de sculpture virtuelle est la gestion de la topologie : les changements sont impossibles avec les techniques de FFD et requièrent l'édition du polygone de contrôle, interne à la représentation, pour les surfaces de subdivision.

Ainsi, nous nous sommes orientés vers une représentation volumique discrète exploitant les techniques associées aux surfaces implicites. Cette représentation permet l'édition *en apparence* directe de la surface affichée, et gère de manière transparente des changements de topologie quelconques. Un autre avantage de cette approche *volumique* est que la surface obtenue définit un volume, i.e. elle possède un intérieur bien défini. En particulier, elle ne présente pas d'auto-intersections comme cela peut se produire avec les déformations FFD ou les surfaces de subdivisions. Ce point est important pour plusieurs domaines d'utilisation des surfaces produites, comme par exemple l'impression $3D$ ou les détections de collisions / calculs d'animation.

2 Interaction avec le modèle

Dans cette partie, nous nous intéressons aux moyens d'interaction, c'est-à-dire aux techniques et outils permettant d'enrichir la perception qu'a l'utilisateur de l'objet *sculpté*. Après un bref survol des différents matériels existants, aussi bien pour l'aspect visuel que pour l'aspect positionnement d'outils et perception tactile et / ou haptique, nous aborderons quelques articles exploitant certains de ces dispositifs.

2.1 Les périphériques

Loin d'être une liste exhaustive des produits et technologies de réalité virtuelle, l'objectif de cette partie est de montrer l'éventail des posssiblités offertes par les technologies existantes et susceptibles d'enrichir une éventuelle métaphore de sculpture virtuelle. Cette partie est donc assez *mouvante* et dépendante des technologies et des modes émergentes. Le but ici n'est pas de désigner *la* configuration idéale pour la sculpture (car elle n'existe pas), mais d'aider à choisir dans un cadre d'utilisation donné une configuration adaptée.
Par exemple on ne travaillera pas sur les mêmes objets en étant debout devant un écran de 10 mètres de large avec un gant de données à main levée, que devant un écran avec une souris et un clavier. En effet, la dimension de l'affichage peut influencer la taille des objets construits, et aussi varier selon le nombre de personnes assistant au travail. De même, selon la précision souhaitée, il peut être plus facile de ne pas travailler à main levée, et selon la durée de ces opérations, la station debout ou la fatigue de la main levée peuvent devenir un problème.
Tous ces critères sont à prendre en compte selon le schéma type d'une session de travail. En tenant sans doute également compte des aspects financiers, il est alors possible d'aboutir à *la* configuration optimale.
Nous nous basons ici sur des états de l'art issus de [Kal93], et de quelques sites indépendants, universitaires ou publicitaires.

Visualisation, affichage

R.S. Kalawsky présente dans [Kal93] une classification des différentes technologies d'affichage existantes en 1993 (figure 1.78) et évoque dans ce livre les principes de ces technologies. La clas-

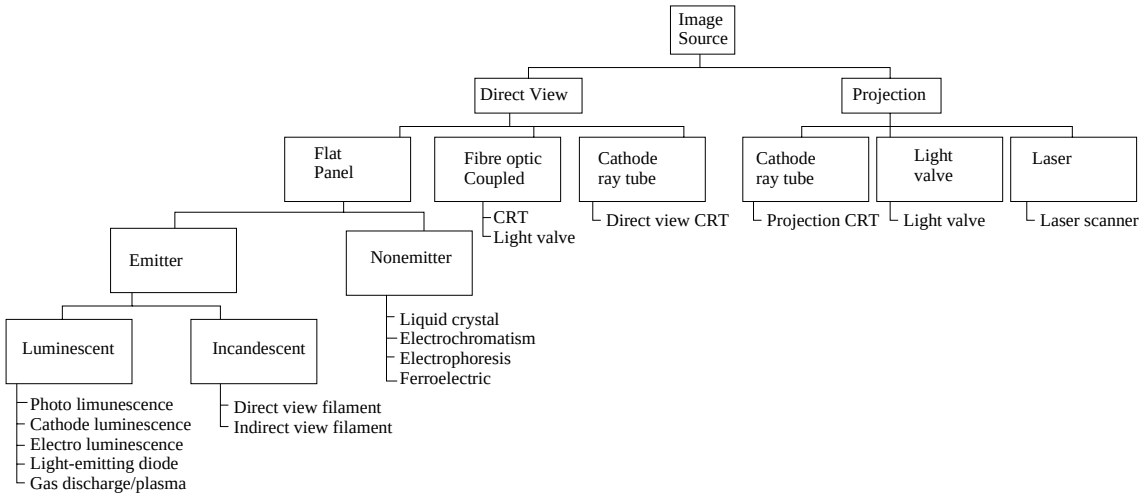


FIG. 1.78: Inventaire des procédés d'affichage répertoriés par R.S. Kalawsky [Kal93]

sification émergeant de [Lab97] et [Lan97] est plus orientée vers l'utilisateur et fait apparaître trois catégories selon la *relation* de l'affichage à l'utilisateur :

1. les visiocasques, aussi appelés HMD pour Head (ou Helmet) Mounted Display. L'utilisateur porte sur sa tête le dispositif d'affichage.
2. les afficheurs immersifs, ou SID pour Spatially Immersive Display. L'utilisateur est immergé dans la scène. Cette catégorie regroupe les dispositifs comme des dômes ou des écrans géants offrant un grand champ de vision et généralement destinés à plusieurs observateurs simultanés.
3. *les autres*, ou VMD pour Virtual Model Display, qui sont des affichages d'extension plus limitée, généralement sur une table ou assimilé comme une tablette graphique, un moniteur, un tableau ...

En théorie, les visiocasques (voir Figure 1.79) représentent les meilleurs dispositifs d'affichage car ils permettent d'afficher une image différente pour chaque œil, donc de restituer une sensation de stéréovision. Les corrections optiques permettent également d'offrir un grand champ de vision et de s'affranchir des contraintes de fusion convergence liées aux systèmes à écran. Nous ne détaillerons pas ici la théorie de la génération d'images stéréo, en particulier les contraintes attachées à leur génération à partir d'écrans, comme la disparité entre la fusion et la convergence. On pourra se reporter pour plus de détails sur ces points à des ouvrages généraux sur la réalité virtuelle [Kal93, PF01b] ou des sites web comme [Bou97] par exemple.

Malheureusement, la théorie demeure loin de la pratique. Les casques existants (voir Figure 1.79) ont des résolutions assez faibles, et un champ de vue assez restreint. Des dispositifs plus évolués (i.e. meilleurs angle de vision et résolution) existent, mais les prix demeurent prohibitifs. Le poids et l'encombrement sont aussi des facteurs limitants sur de tels systèmes reposant sur la tête de l'utilisateur. L'affichage des images générées requiert également un câble, qui ajoute des limitations sur le confort et l'utlisation prolongée.

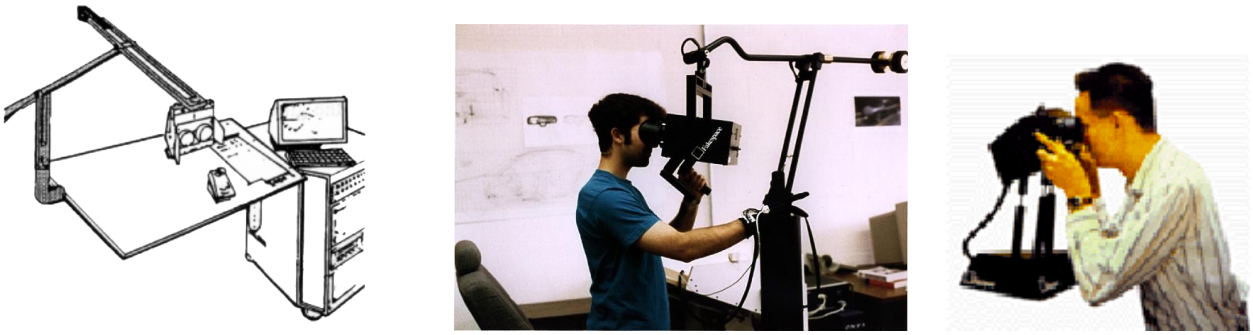
Un autre inconvénient de ce système est lié à son avantage, c'est-à-dire l'immersion complète de l'utilisateur. Cette immersion impose des contraintes fortes sur la fréquence de génération d'images pour ne pas provoquer de malaise ou inconfort. L'autre aspect plus psychologique concerne l'isolement, qui empêche toute collaboration ou interaction avec l'environnement réel avoisinant etavec son

(a) (b) (c) (d)

FIG. 1.79: (a) le visiocasque semi-transparent de *i-glasses !* (image extraite de `www.vio.com`). (b) le Proview 50, version montée sur casque de *Kaiser Electro-Optics* offrant une résolution de $(3 \times 640) \times 480$ par oeil avec un champ de vue de $50^o \times 30^o$, (c) la version commerciale du ProView 40 et (d) le SIM EYE 40, basé sur des afficheurs à tube cathodique (CRT) contrairement aux ProView basés sur des afficheurs à cristaux liquides (LCD) (source `www.cts.com/browse/keo`).

propre corps.
Des alternatives existent (domaine de la Réalité Augmentée) faisant appel à des visiocasques semi-transparents ou des caméras posées sur le casque pour fusionner une image de l'environnement avec l'image calculée. Il est ici possible d'incruster des objets virtuels dans une scène réelle. L'utilisateur dispose de repères dans le monde réel, mais la complexité de l'étalonnage pour faire correspondre les deux scènes (réelles et virtuelles) fait encore l'objet de recherches [GDG01, GG02].

Dans tous les cas, les problèmes de réglages optiques, de latence entre les mouvements et l'image affichée, et de poids sont sources de fatigues et peuvent limiter le confort et l'utilisation de ces systèmes.



(a) (b) (c)

FIG. 1.80: (a) l'*afficheur stéréoscopique à tube cathodique avec contrepoids* de NASA Ames (1985) qui permettait d'atteindre 400 lignes de résolution avec des afficheurs *CRT* monochromes et un champ de vue de $120^o \times 60^o$ (image et données extraites de [Kal93]). (b) le BOOM (Binocular Omni-Orientation Monitor) de Fakespace (1988). (c) Fakespace Push.

Des alternatives palliant notament la résolution et le poids des visiocasques ont été proposées, comme les *Boom* et *Push* de Fakespace (voir figure 1.80). La Figure 1.81 montre des extensions plus ou moins exotiques de ces concepts. Ces deux limitations tendent toutefois à s'estomper avec la miniaturisation des afficheurs permettant un poids toujours plus réduit pour une meilleure résolution : par exemple, les *Datavisor* de *n-Vision* qui atteignent une résolution 1280×1024 pixels par oeil, ou les visiocasques de KEO (Kaiser Electro Optics) qui utilisent une technique de *pavage 2D* d'écrans LCD : le site `www.cts.com/browse/keo` présente un visiocasque offrant 6 pavés (écrans à résolution VGA) par oeil, soit une résolution de $(3 \times 480) \times (2 \times 640)$ et évoque la construction d'un prototype

(a) (b)

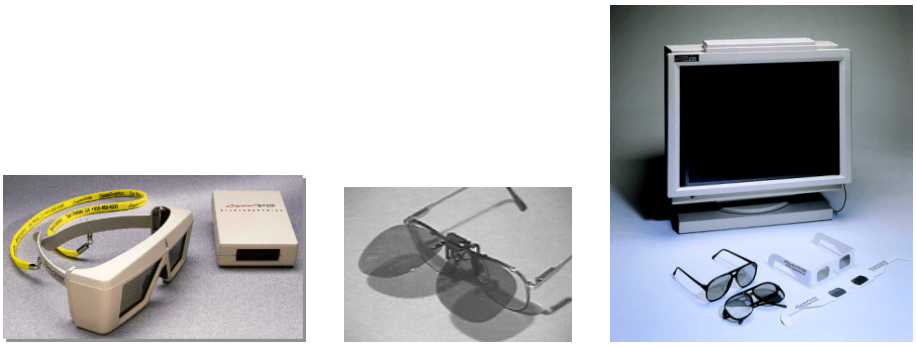
FIG. 1.81: (a). Flostation de Flogiston (image extraite de `www.flogiston.com`). (b) exemple d'utilisation d'un BOOM en mode *mains libres* (image extraite du site de la société Fakespace.

à 15 pavés par oeil.

La nouvelle technologie des VRD (Virtual Retinal Display) apparue en 1993 au HITLab (Human Interface Technology Lab de l'université de washington (`www.hitl.washington.edu`) utilise des diodes laser dont les faisceaux sont modulés pour balayer la rétine et y projeter directement l'image calculée. Cette technologie présente plusieurs avantages parmis lesquels une faible consommation électrique, des luminosités plus élevées qu'avec les moniteurs classiques, un spectre de couleurs plus large, pas de rémanence autre que celle de la rétine elle-même, et une résolution qui n'est plus limitée par une grille de phosphores ou de cristaux liquides, mais uniquement par les systèmes optiques.
Ces technologies sont encore en développement, mais quelques systèmes commencent à apparaître comme par exemple le *Spectrum* de Microvision (voir `www.mvis.com`).

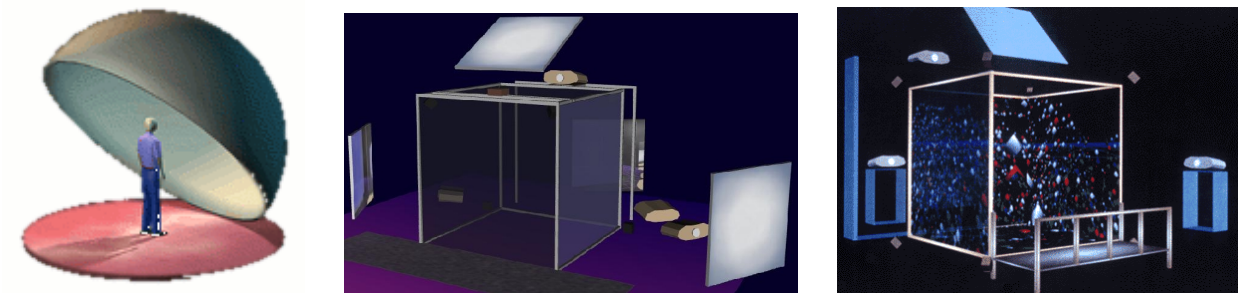
Dans la plupart des dispositifs autre que les visiocasques, la stéréovision est assurée en séparant les images destinées à chaque oeil. La séparation peut être assurée **temporellement**, on parle alors de stéréovision **active**. Les techniques les plus courantes font appel à des lunettes à cristaux liquides qui permettent d'obturer séquentiellement chaque œil. Elles requièrent une synchronisation avec le générateur / afficheur d'images, généralement assurée par une transmission infrarouge. Se posent alors les problèmes de placement et de gestion des occultations des émetteurs. Il est aussi nécessaire d'avoir un récepteur, et donc une pile dans les lunettes. Cela impose un poids, et un coût plus important pour chaque paire de lunettes. Un autre inconvénient de ce dispositif est la division de la fréquence image. On voit un exemple de ce dispositif Figure 1.82.(a). Un autre principe de séparation utilise des filtres polarisants, on parle de stéréovision **passive**. Le principe consiste à utiliser une source d'image polarisée différemment pour chaque œil. La séparation se fait ici aussi par des lunettes qui sont simplement des filtres polarisants (voir par exemple Figure 1.82.(b)). Ces lunettes ne nécessitent pas de dispositif de synchronisation, donc pas de piles. Elles sont plus légères et moins coûteuses. Dans le cas d'un système d'affichage projeté, il est possible de doubler le nombre de projecteurs utilisés et ainsi de ne pas diviser la fréquence image. La luminosité est également plus forte qu'avec les dispositifs actifs. Des dispositifs de polarisation actifs, par exemple pour des écrans existent également (comme par exemple le Zscreen de stereographics, voir Figure 1.82.(c)).

La catégorie des affichages **SID** (Spatially Immersive Display) désigne des dispositifs de projection sur de grands écrans. Ils sont généralement utilisés pour plusieurs observateurs simultanés. On trouve plusieurs variantes selon la forme de l'écran et le système de projection (frontale ou rétro-projecté). Par exemple, le CAVE développé à l'EVL (Electronic Visualization Laboratory, University of Illinois) et présenté à SIGGRAPH'93 (voir [CNSD93] et Figure 1.83.(b)). Le système uti-



(a) (b) (c)

FIG. 1.82: (a) lunettes obturantes à cristaux liquides *Crystal Eyes* de *Stereographics*, photographiées avec l'émetteur infrarouge pour la synchronisation avec l'affichage (image extraite de `www.stereographics.com`). (b) exemple de filtre polarisant amovible sur des lunettes classiques (c) lunettes polarisantes avec un filtre devant un écran (dispositif *Zscreen* de *Stereographics*).



(a) (b) (c)

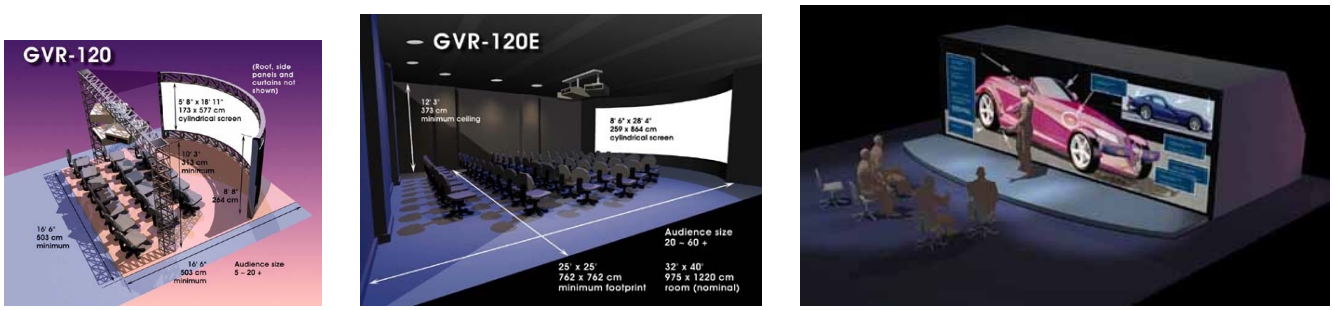
FIG. 1.83: (a) principe du visionDome d'Alternate Realities Corporation (b) schéma du CAVE, illustration de Milana Huang extraite du site de l'Electronic Visualization Laboratory (`www.evl.uic.edu`). (c) schéma extrait du site de Pyramid Systems (fermé depuis, et commercialisation reprise par *Fakespace Systems* `www.fakespace.com`).

lise une configuration en cube, avec un affichage stéréoscopique. Le système initial comportait trois murs avec un sol et un plafond. Des variantes proposent un nombre réduit ou supérieur d'écrans. De tels systèmes sont commercialisés par les sociétés *Barco*[3] et *Fakespace*[4] (voir figure 1.83.(c)). Des systèmes constitués de 3 écrans pouvant former des angles quelconques, entre l'écran plat et les 3 faces d'un CAVE restreint, sont également commercialisés (*MoVE* de *Barco* et *RAVE* de *Fakespace*).

Un autre exemple est le *visionDome* de ARC (Alternate Realities Corporation) exposé pour la première fois lors de SIGGRAPH'92. Le principe est de projeter les images sur un dôme de 360^o par 160^o immergeant complètement l'utilisateur (voir figure 1.83.a). Le dispositif complet comprend un dôme et des optiques correctrices pour le projecteur (*light valve*).
Une autre branche regroupe les *salles de réalité virtuelle*, disposant un grand écran cylindrique ou plan éventuellement rétroprojeté devant des spectateurs. Pour exemple, la Figure 1.84 présente un éventail de solutions de ce type proposées par la société Panoram décomposé en gamme GVR pour Group Virtual Reality (désignant des écrans courbes, sans stéréo) et gamme PanoWall (désignant des écrans plats, rétro-projetés avec stéréo).

La catégorie des **VMD** (Virtual Model Display) comprend les dispositifs utilisant des écrans plus petits et destinés à un nombre plus faible d'observateurs. Par exemple le Responsive Workbench

[3]Barco a racheté TAN le 8 août 2002
[4]Fakespace et Pyramid Systems ont fusionné en 1999, annonce faite lors du Siggraph



(a) (b) (c)

FIG. 1.84: Spatially Immersive Display de Panoram Technologies, pour la réalité virtuelle de groupe : (a) une version transportable, le GVR-120 présentant une image de $5,77 \times 1,73$ mètres, (b) une version *fixe*, le GVR-120E présentant une image de $8,64 \times 2,59$ mètres. (c) le PanoWall 90 présentant une image de $7,52 \times 2,29$ mètres (images extraites de `www.panoramtech.com`).

développé par W. Krüger au GMD (German National Research Center for Information Technology) et actuellement commercialisé par la société *Barco*. Le principe est d'utiliser un projecteur pour former des images sur une table (figure 1.85.a). Les procédés stéréoscopiques (lunettes) permettant de

(a) (b) (c) (d)

FIG. 1.85: (a) le TAN Responsive Workbench de Barco (image extraite de `www.barco.com`) (b) Immersive Workbench de Fakespace (c) ImmersaDesk de l'EVL commercialisé par Pyramid Systems (image extraite du site de l'EVL `www.evl.uic.edu`). (d) PanoDesk36 de Panoram Technologies (source `www.panoramtech.com`)

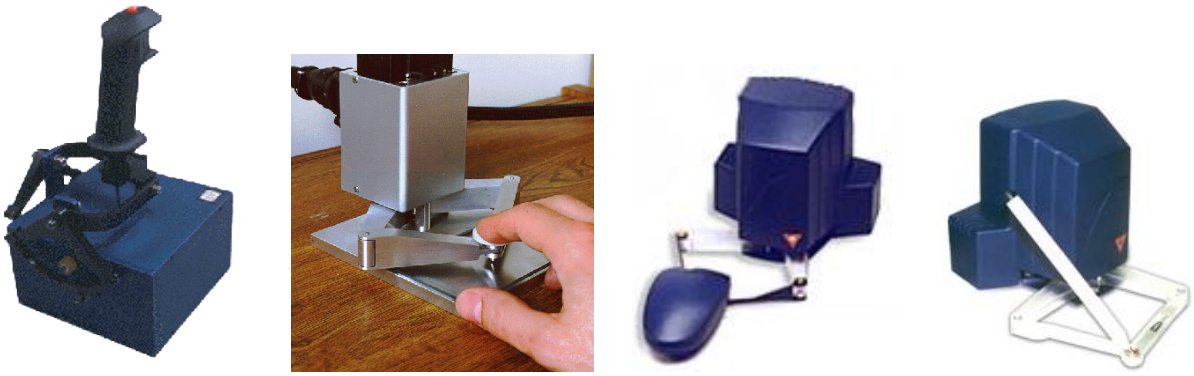
donner la sensation que l'image *sort* de la table. L'Immersive Workbench commercialisé par Fakespace (figure 1.85.b), reprend le même principe. Présentant une orientation différente, l'ImmersaDesk développé à l'EVL (Electronic Visualization Laboratory) est commercialisé par Fakespace (figure 1.85.c). Plusieurs variantes existent ici aussi selon la taille et le nombre d'écrans ; par exemple, les modèles *TAN Holobench* et *Consul* de *Barco* utilisent deux écrans diposés en L.

Dans cette catégorie, on peut également ranger un nouveau type d'afficheurs, dits *autostéréo*, qui permettent à un ou plusieurs observateurs d'observer un écran en stéréovision sans lunettes dédiées. Plusieurs techniques existent pour séparer les images provenant à chaque œil. Un premier prototype proposé par K. Perlin dans [PPK00] consiste à disposer un filtre actif devant un écran pour masquer certains pixels. Il nécessite le suivi très précis de la position des yeux de l'observateur. La génération de paires stéréo et leur entrelacement sur l'écran dépend également de ces positions, et doit donc être adaptée à chaque image.
Une autre solution proposée par P. Allio (voire `www.alioscopy.com`) utilise un réseau optique collé sur l'écran pour séparer les tranches de pixels. Ce dispositif ne nécessite pas de suivi de la tête et supporte plusieurs observateurs simultanés. Il fournit trois paires d'images stéréos contiguës.

Manipulation, édition

Nous essayons dans cette partie de classer les différents types d'interfaces de manipulation que nous avons pu recenser en fonction des informations qu'elles fournissent (positionnement 2*D*, 3*D* ou plus) et éventuellement du type de contraintes sur les mouvements de l'utilisateur (retour d'effort) qu'elles offrent.

On trouve ainsi à une extrémité de notre classement l'interface standard existant sur toute machine actuelle : une souris 2*D*, avec 1 à 3 boutons.
Des variantes basées sur un dispositif de *Pantograph* (voir Figure 1.86.(b)) développé à l'université



(a) (b) (c) (d)

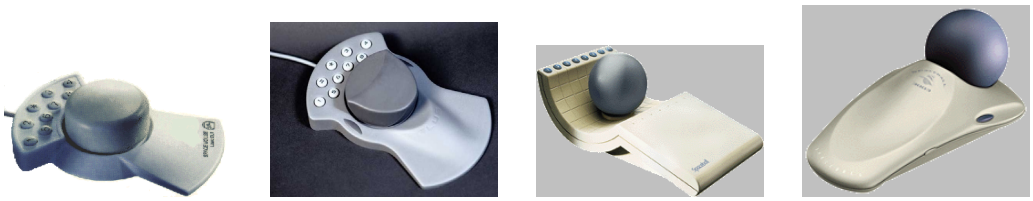
FIG. 1.86: (a) le joystick à retour d'effort *Impulse Engine 2000* de *Immersion Corp.* (b) la version développée à l'université Mc Gill (c) version commerciale munie d'une souris (c) version commerciale munie d'un stylet.

de McGill proposent des souris 2*D* enrichies avec des retour d'effort. Ces interfaces étaient originellement destinées à aider les personnes à vision déficiente pour évoluer dans un environnement fenêtré (associer des gravités aux fenêtres, simuler leurs bords, etc . . .).

Les **joysticks** interface de jeu par excellence. Des versions incorporant un retour d'effort ont été proposées, comme par exemple l'Impulse Engine ou Impulse Stick d'Immersion (voir Figure 1.86.(a)) ou bien la version *SideWinder* de Microsoft. Bien que des recherches fondamentales sur le rendu haptique aient été menées avec ce type d'interface (par exemple, M. Minsky utilise dans [MOyS$^+$90] pour ses études sur la perception des surfaces), ce positionnement relatif uniquement 2*D* nous a semblé d'emblée peu exploitable dans un contexte d'interaction directe avec une surface, aussi bien pour l'édition que l'exploration.

Les **souris 3D** fournissent un positionnement relatif 6*D* (i.e. les trois translations et trois angles de rotation) pour un encombrement et une utilisation assez proche des souris 2*D*. Plusieurs dispositifs existent (voir Figure 1.87). Ces *souris étendues* sont assez robustes et de faible coût. Elle permettent un positionnement et une manipulation aisés. Dans le cadre d'une interaction à deux mains, elles conviennent bien pour le déplacement de la scène. Par exemple, L. Moccozet and P. Kalra [MK93] font référence à une métaphore de sculpture classique à deux mains, appelée *ball and mouse metaphor* qui consiste à positionner l'objet sculpté en position et orientation avec une souris 3-D et à modifier cet objet avec un outil contrôlé par une souris 2-D.

Les **tablettes graphiques** permettent de dessiner avec un stylet comme sur une feuille de papier ordinaire. Des variantes existent selon la richesse des informations d'orientation et de pression



(a) (b) (c) (d)

FIG. 1.87: (a) la Space Mouse et (b) la Space Mouse Plus de *Space Control* (images extraites de `www.spacemouse.com`). (c) la SpaceBall 2003 et (d) la SpaceBall 3003 de Spacetec IMC (images extraites de `www.spacetec.com`).

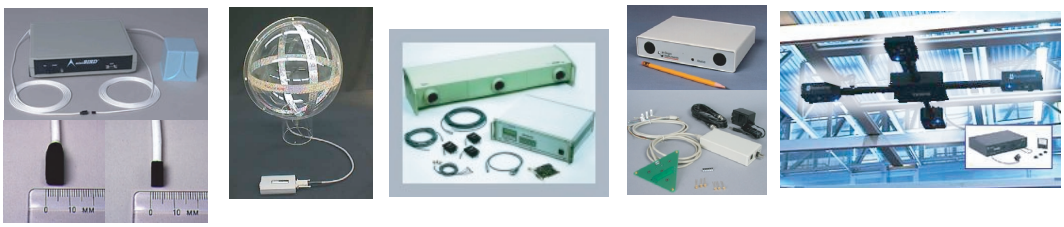
collectées. Pour exemples, *Wacom* (`www.wacom.com`) propose des produits plus ou moins évolués, depuis les *Graphire2* avec 512 niveaux de pression, jusqu'aux divers *Intuos2* avec 1024 niveaux de pression, la mesure de l'orientation du stylet utilisé et dont la taille varie du format A6 au A3 (voir Figure 1.88.(a)).
Des extensions intéressantes existent qui combinent la tablette avec l'écran, offrant ainsi la possibilité

(a) (b) (c)

FIG. 1.88: Exemples de tablettes. (a) tablette *Intuos2* de Wacom (b) tablette combinée à un écran LCD *Cintiq* de Wacom et (c) un *tablet PC* le modèle *Stylistic 3500* de Fujitsu

de dessiner directement comme sur une feuille de papier (voir Figure 1.88.(b)).
D'autres extensions incorporent même la machine complète avec l'écran et la tablette dans un ensemble appelé *tablet PC* (voir Figure 1.88.(d)). On trouve aussi sur le site Microsoft une version dédiée de WindowsXP pour les *tablet PCs* en partenariat avec les grands constructeurs (pour exemple, citons Acer, Fujistu, HP, Toshiba, VIA Technology ou ViewSonic, NEC) qui semble assoir la pérennité de ces appareils.

Les **capteurs de position** peuvent également être utilisés comme pointeurs 6*D* pour l'édition. Ces capteurs sont de toute façon requis dans la plupart des dispositifs pour, par exemple, connaître la position des yeux de l'observateur et être capable de construire une paire d'images stéréo adéquate.
Ici aussi, plusieurs technologies existent. Les principales étant magnétiques, à ultrasons ou optiques. Les plus répandues à l'heure actuelle sont les solutions magnétiques, sans doute à cause de leur coût sensiblement plus faible que pour les autres solutions plus récentes. Ces solutions magnétiques sont toutefois très contraignantes envers l'environnement qui ne doit pas contenir de pièces métalliques, ou plus largement, d'éléments perturbant le champ magnétique. Ces technologies nécessitent des corrections pour étalonner les déformations du champs dans une installation donnée (voir par exemple les pages 175 à 201 de la thèse de G. Zachmann [Zac00]). La Figure 1.89 montre les capteurs magnétiques *miniBird* d'Ascension Technology Corporation (`www.ascension-tech.com`) et le *Long Ranger* de Polhemus (`www.polhemus.com`).
Une version utilisant des leds infrarouges est proposée par *Northern Digital Inc.* (`www.ndigital.com`), dont on voit le système Polaris dans la Figure 1.89.(c). L'autre système *Optotrack* proposé par cette même société est plus connu. Un système reposant sur les mêmes principes (infrarouge) proposé par *Origin Instruments* (`orin.com`) utilise des leds lumineuses synchronisées (voir Figure 1.89.(c)) ce



(a) (b) (c) (d) (e)

FIG. 1.89: Exemples de capteurs de position. (a) les miniBirds 800 et 500 d'*Ascension Technology Corporation* ; (b) le Long Ranger de Polhemus ; (c) le dispositif infrargouge *Polaris* de Northern Digital (d) le capteur Dynasight d'Origin Instruments avec sa cible active composée de 3 leds infrarouges ; (d) le dispositif *IS600* d'Intersense. (images extraites des site webs respectifs)

qui permet de suivre des cibles identifiées. Le système propose aussi l'utilisation de cibles passives. Un exemple de système à ultrasons est l'*IS600* d'Intersense (`www.isense.com`) montré à la Figure 1.89(e).
Dans la plupart des cas, des solutions sans fils utilisant des marqueurs passifs existent.

Les **bras articulés** sont un autre moyen de mesurer une position et une orientation. De tels dispositifs sont exploités notament pour la numérisation d'objets, comme par exemple avec le *MicroScribe* d'Immersion. Un aspect qui nous intéresse plus ici concerne la possiblité d'agir sur les mouvements du bras, c'est-à-dire de créer des *forces* pouvant éventuellement reproduire la sensation de toucher un objet virtuel avec le stylet manipulateur.

Un des premier systèmes commercial fut proposés par *Sensable* (`www.sensable.com`), société fondée en 1993 par T. Massie et K. Salisbury. Il reprend les développements effectués au MIT Artificial Intelligence Laboratory. La gamme de produits s'est depuis enrichie pour offrir maintenant 5 différents modèles. Sensable commercialise également un système de sculpture virtuelle appelé *FreeForm*, sur lequel nous reviendrons au long du document. On voit par exemple le plus petit de la gamme, le *PHANTOM Desktop* en Figure 1.90.(a). Il mesure la translation et la rotation du stylet et permet de générer une force en translation. Le modèle présenté sur la Figure 1.90.(b) permet de plus de produire un couple, soit un *retour d'effort* 6*D*.

(a) (b) (c) (d)

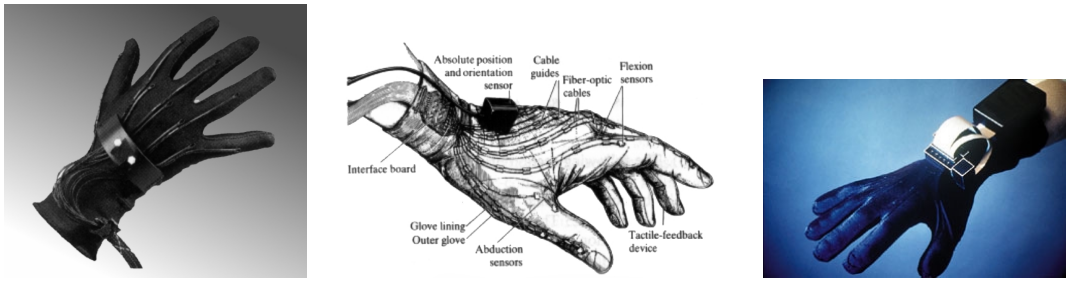
FIG. 1.90: (a) le modèle de bras articulé à retour d'effort *PHANTOM Desktop* et (b) *1.5/6DOF* de *Sensable* ; (c) le modèle *Virtuose* de *Haption* ; (d) le modèle Delta 3-DOF de *Force Dimension*

D'autres systèmes commencent à *sortir des laboratoires*. Par exemple, la société *Haption* commercialise un dispositif très similaire en présentation, mais reposant sur des technologies (moteurs) différents développés au CEA (`www.haption.com`, voir Figure 1.90.(c). La société *Force Dimension* (`www.forcedimension.com`) propose un dispositif de retour d'effort découplant les translations et les rotations. Son système, le *Delta* (voir Figure 1.90.(d)) existe en version 3*D* et 6*D*, cette dernière

bénéficiant de contrôles supplémentaires sur les rotations.

La société Sensable propose également des variantes du système phantom munies d'une sorte de *dé à coudre* à la place du stylet. Ce dispositif avec plusieurs bras permet de contrôler plusieurs doigts de la main.

Les **gants de données** sont issus historiquement du DataGlove (voir Figure 1.91 (a) et (b)) développé par VPL. Actuellement commercialisés, on trouve par exemple le *5th Glove* de *General Reality* (voir Figure 1.91.(c)) ou encore le *CyberGlove* de *Virtual Technologies* et maintenant commercialisé par Immersion (voir Figure 1.92). Les gants fournissent des informations de flexions des doigts via des technologies basées sur les fibres optiques ou plus généralement sur des mesures de propriétés variant avec la flexion ("*proprietary resistive bend-sensing technology*" des *CyberGloves*). La position de la main est quant à elle connue via des capteurs de position classiques, magnétiques



(a) (b) (c)

FIG. 1.91: Le VPL-Dataglove (a) et son schéma (b) (images extraites de [Kal93]). (c) le *5th glove* de *General Reality* (image extraite de `www.genreality.com`).

dans la plupart des cas. La version la plus simple mesure juste la configuration de la main. Il existe

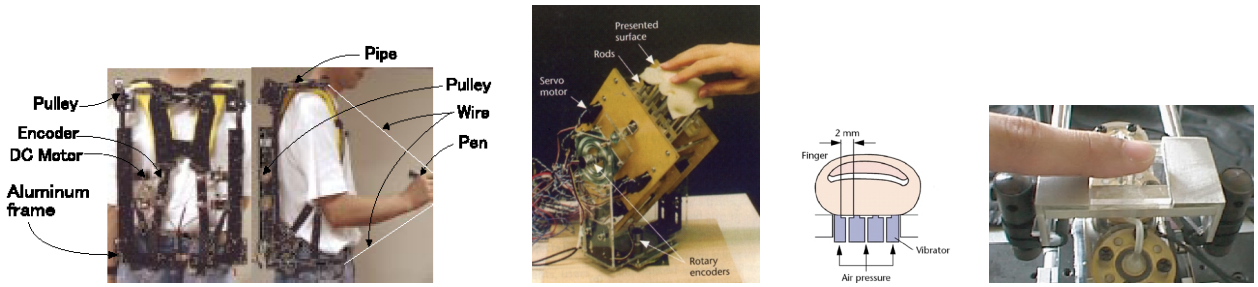
(a) (b) (c)

FIG. 1.92: La gamme de gants de données d'Immersion (a) le *CyberGlove*, (b) le *CyberTouch*, (c) le *CyberGrasp* et (d) le *CyberForce2* (images extraites de `www.immersion.com`).

des variantes proposant un retour tactile avec des vibrations (voir Figure 1.92.(b)) ou bien permettant de bloquer les mouvements des doigts (par exemple le *CyberGrasp* de la Figure 1.92.(c)). Ce dernier peut être combiné avec un dispositif permettant en plus de contrôler les mouvements du bras (comme montré à la Figure 1.92.(d)).

Dans le cas où l'utilisateur est immergé avec un affichage par projection, de type SID ou VMD comme une salle de réalité virtuelle ou un *Responsive Workbench*, les solutions précédentes de retour d'effort à base fixe peuvent se révéler délicates à utiliser. En effet, elles conviennent bien si les dispositifs peuvent être masqués ou hors de la vue, mais sont gênantes lorsque ces derniers sont dans le champ de vue. Des technologies de retour d'effort par câbles, qui peuvent être invisibles si suffisament fins, ou bien carrément portés par l'utilisateur existent pour pallier ces inconvénients. La Figure 1.93.(a) montre par exemple un *HapticGear* présenté par M. Hirose et al. dans [HOYK99]. C'est un dispositif utilisant des câbles avec des codeurs optiques pour connaître la position du stylet, et des

moteurs pour la commander.



(a) (b) (c) (d)

FIG. 1.93: (a) le dispositif *HapticGear* de retour d'effort portable (image extraite de [HOYK99]) (b) prototype de stimulation tactile utilisant des actuateurs mécaniques présenté dans [HH95] (images extraites de l'article). (c) et (d) principe et prototype de stimulateur couplant la pression d'air et les vibrations pour stimuler différents récepteurs de la peau (images extraites de [AYS98]).

Une direction de recherche émergente concerne la simulation du toucher (*tactile feedback*). Aucun dispositif commercial n'existe encore, à part les stimulateurs vibrants du CyberTouch. Des dispositifs à base de jets d'air ou d'actuateurs mécaniques sont en développement dans les équipes de recherche. Par exemple, la figure 1.93 (a) et (b) montre le dispositif mécanique de K. Hirota et M. Hirose. Les Figure 1.93 (c) et (d) représentent le principe et un prototype couplant à la fois de l'air sous pression et des vibrations pour activer des récepteurs de la peau. Ce dispositif sert également à étudier sur des sujets l'influence de ces stimulations, pour progresser dans la simulation des sensations tactiles.

Après cet aperçu des technologies axé sur la perception et l'interaction, nous allons illustrer quelques exemples d'utilisation de ces technologies. Naturellement, il est délicat de trouver un ordre de parcours ou un découpage avec ces deux critères entremêlés. Comme dans la partie consacrée aux représentations, nous proposons d'illustrer ces thèmes à travers un cheminement entre des articles marquants pour balayer les combinaisons possibles. On tâchera toutefois dans la première partie de mettre l'accent sur le type de périphérique (dimensionnalité, retour haptique, ...) utilisé. Dans la seconde, on dégagera plus l'intérêt des dispositifs de visualisation mis en place.

2.2 Utilisation d'une entrée $2D$

Bien que ne nous intéressant pas directement, des applications de *design* grossier et exploratoire utilisant une interface $2D$ de type souris ou stylet avec tablette on été proposées. Ces interactions de type *crayon-papier* sont très proches de la démarche utilisée en phase préparatoire ou de discussions pour faire des croquis. A la différence d'un croquis, on trouve ici en résultat une forme initiale tridimensionnelle.

SKETCH : An interface for sketching 3d scenes

Zeleznik et al. proposent en 1996 [ZHH96] un système qui génère à partir de tracés sur l'écran les primitives correspondantes (comme des boîtes, des sphères, des cônes, ...). Une gestuelle codifiée permet ensuite des les positionner ou d'effectuer des opérations pour les combiner (addition, soustraction, regroupement).

Une souris à trois boutons est utilisée en entrée, avec éventuellement une touche du clavier. Le premier bouton permet de dessiner des courbes. La sémantique utilisée avec les gestes et ce bouton est détaillée dans la Figure 1.94.

A leur création, les primitives sont positionnées selon un ensemble de règles simples. Chaque pri-

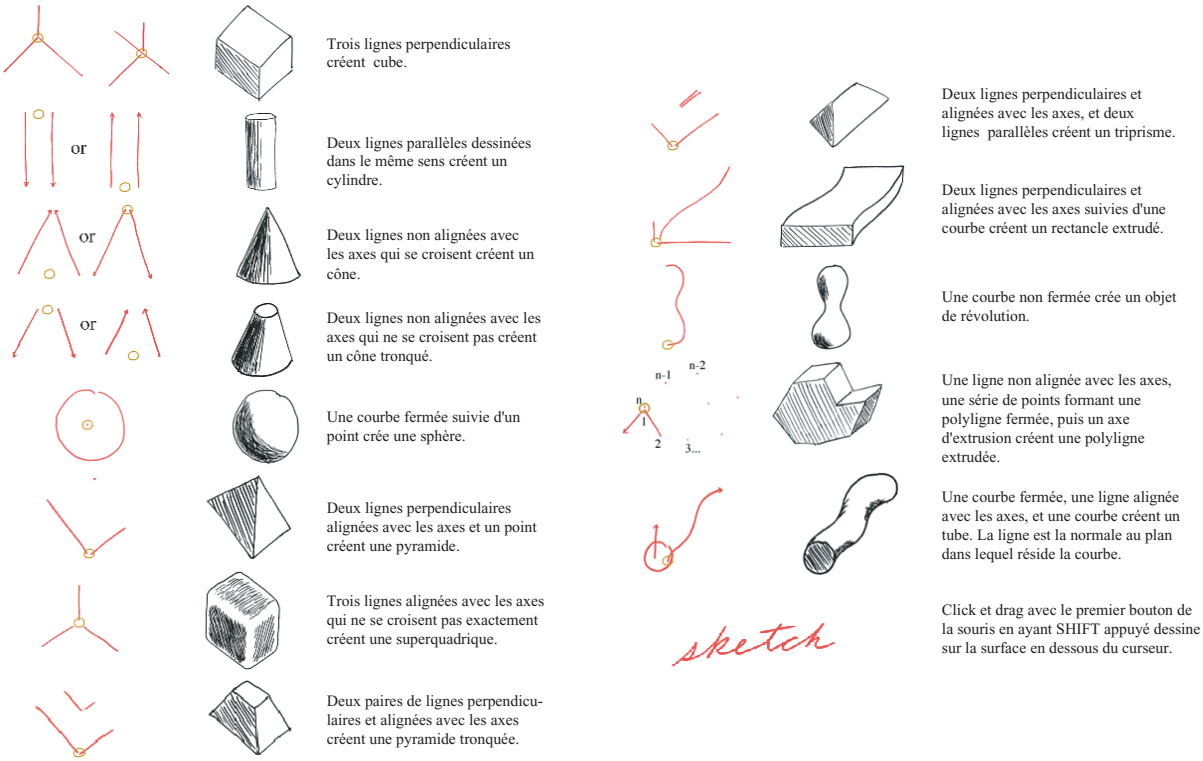


FIG. 1.94: Liste des gestes prédéfinis dans SKETCH pour créer des primitives (figure extraite du site `www.cs.brown.edu/research/graphics/research/pub/papers/sig96-sketch/sig.html`).

mitive possède un point d'intérêt qui permet de la positionner par rapport à celles existantes. La direction des gestes générant la primitive permet également de spécifier des opérations CSG : si un des traits pénètre une primitive existante, la primitive générée lui est soustraite. Le troisième bouton sert à modifier les paramètres de caméra.

Concernant l'édition, un tracé précise la transformation souhaitée, et l'interaction se fait ensuite avec le bouton central. Il est possible de changer la taille en traçant deux traits de sens opposés ou de préciser le placement en dessinant une ombre sous un objet sélectionné. Enfin, plusieurs déplacement contraints sont possibles en précisant un axe de rotation ou deux axes pour le cas d'une translation dans un plan.

Teddy : A Sketching Interface for 3D Freeform Design

Si l'on pouvait reprocher à SKETCH d'imposer une codification complexe de la gestuelle et de créer des objets de forme *simples* ou d'être incapable de créer des formes *organiques*, la solution proposée par T. Igarashi et al. en 1999 [IMT99] résoud quelques unes de ces limitations.

En effet, l'interface est limitée à un stylo ou une souris à deux boutons (dont un ne sert qu'aux déplacements de caméra), et cinq boutons d'interface qui sont *Init*, *Undo*, *Bend*, *Load* et *Save* (voir Figure 1.95). Plutôt que de détailler les modalités d'interface, on se reportera aux figures qui suivent. Beaucoup d'idées simples permettent de deviner le comportement attendu, et ainsi évitent une interface utlisateur classique (menus, boutons, cases à coccher, ...). Le raturage permet selon le contexte d'effacer une courbe dessinée sur la surface ou de lisser une zone de la surface (voir Figure 1.96). L'utilisation d'un bouton est nécessaire pour entrer dans le mode déformation (voir Figure 1.97 à gauche). La construction de la surface à partir d'un contour (Figure 1.97.a) passe par le calcul d'une

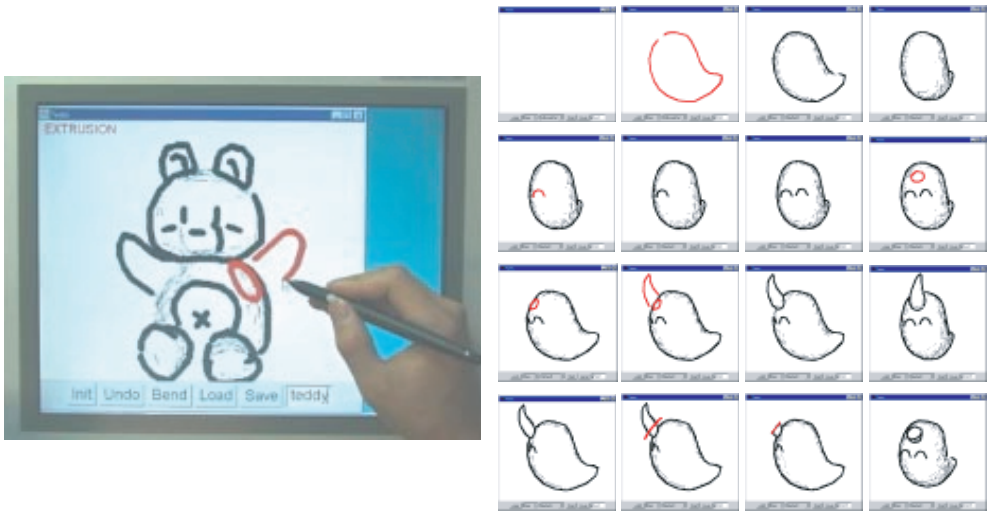


FIG. 1.95: Interface de Teddy. A droite, on voit de gauche à droite et de haut en bas une sucession de de capture écran depuis le démarrage. Dessin d'une courbe, puis inférence de la forme 3D, rotation, dessin d'un oeil sur la surface, dessin d'une courbe pour attacher une corne, dessin de la corne, construction, puis découpe de cette corne (images extraites de [IMT99]).

Creusement selon un contour.

Extrusion selon une courbe. Un click désactive le contour.

Trait de découpe qui produit un contours actif et permet avec un nouveau tracé de prolonger la forme.

Utilisation du gribouillage pour effacer une courbe dessinée sur la surface.

Utilisation du gribouillage pour lisser la surface.

FIG. 1.96: Exemples d'extrusion et de raturage (ou gribouillage) pour effacer des éléments ou lisser la surface (images extraites de [IMT99]).

triangulation de Delaunay contrainte (Figure 1.97.b) pour en extraire l'axe médian (Figure 1.97.c). A partir de là, les triangles extrémaux sont recherchés (Figure 1.97.d) pour obtenir un axe médian simplifié (Figure 1.97.e) permettant de construire une forme 3*D* par élévations.

Teddy a de séduisantes possibilités d'éditions et une interface extrêmement simple. Par contre le système ne permet pas la construction de formes trop compliquées (la topologie des objets est limitée à une sphère ; il est impossible par exemple de construire un tore) ni la construction de plusieurs objets. Les auteurs reconnaissent également la difficulté d'enrichir les opérations sans compliquer les interactions.

Ces deux approches sont très séduisantes, car elles offrent un moyen très intuitif, proche du crayon et du papier utilisés pour les croquis, pour construire rapidement des formes 3*D*. Il n'est pas nécessaire de manipuler des points de contrôle ou positionner précisément des éléments. La facilité d'édition encourage de plus les modifications / essais. L'exploitation d'une gestuelle simple permet de réduire au maximum l'interface, ce qui augmente la sensation de facilité d'utilisation. Dans tous les cas, ces

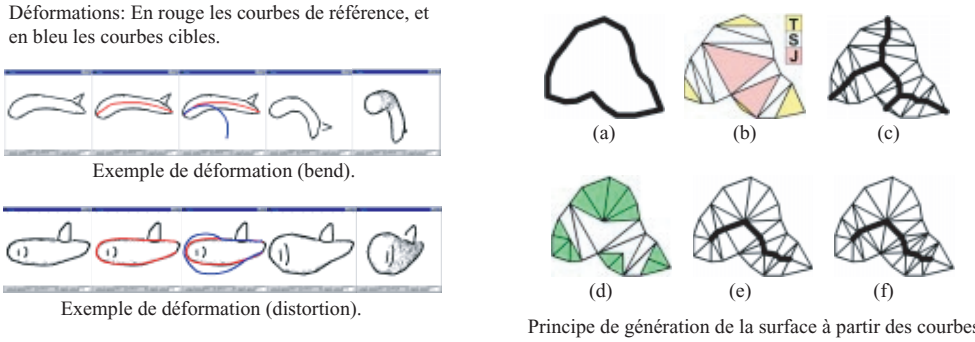


FIG. 1.97: Exemples de déformations, à gauche, et principe de génération des surfaces à partir des courbes à drote (images extraites de [IMT99]).

approches sont délicates à étendre, car l'ajout de fonctionnalité complique les codes d'interaction, et cette codification doit de plus être soigneusement étudié avec les utilisateurs pour *coller* à leur habitudes.

D'autres approches de sculpture que nous avons déjà abordées préfèrent une interaction 2*D*. Par exemple pour la sculpture de bas relief, A. Sourin [Sou01] utilise un stylet avec une tablette graphique sensible à la pression, ce qui semble dans ce contexte, que l'on pourrait qualifier de *sculpture* $2D^{\frac{1}{2}}$, très adapté. Dans un exemple plus proche de notre intérêt, i.e. sculpture d'objet 3*D*, S. Wang et A. Kaufman [WK95] préférent une souris 2*D* à un capteur de position 3*D*, car la souris est selon eux "*plus facile à manipuler*". En outre, l'utilisation d'un capteur de position requiert une détection de collision entre l'outil et l'objet, et comme la position ne serait pas contrainte (pas de retour d'effort envisagé), la pénétration aléatoire de l'outil dans la surface sculptée apparaît trop dérangeante aux auteurs.

2.3 Utilisation d'un capteur de position

C'est pourtant un tel capteur que T. Galyean et J. Hughes [GH91] utilisent pour positionner leurs outils. Le modèle d'outil défini ne permet pas d'utiliser des outils *orientés* (voir section 1.8), ainsi seule la position du capteur est exploitée.

Le capteur renvoie sa position absolue, et les auteurs signalent que malgré une mise en correspondance de l'espace utile du capteur et de l'espace de travail, le bruit des mesures empêche une utilisation précise. La solution mise en place pour minimiser le bruit des mesures consiste à filtrer

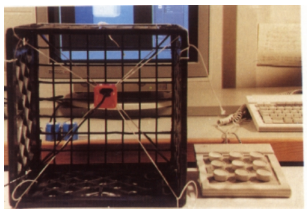


FIG. 1.98: *Poor man force feedback unit* : dispositif utilisé dans [GH91] pour contrôler en *mode relatif* la position des outils (le nom et la figure sont extraits de [GH91]).

les données du capteur, et à proposer un mode **relatif** où l'outil est déplacé incrémentalement, à la manière d'une souris 2D qu'on peut lever et repositionner pour déplacer un curseur sur un écran. L'effet *lever* de la souris 2D est ici rendu par le biais d'un bouton contrôlant la (non-)prise en compte du déplacement. Les incréments sont mesurés par rapport à une position *de repos* donnée par le montage de la figure 1.98 : le capteur est maintenu au centre d'une boîte par des élastiques et demeure ainsi

dans une zone de bonne stabilité pour les mesures du capteur.

2.4 Utilisation d'un stylet avec retour d'effort

A Haptic Interaction Method for Volume Visualization

Cet article, que nous avons évoqué en 1.8 s'intéresse au rendu haptique de surfaces définies comme isosurfaces d'un champ scalaire échantillonné sur une grille régulière.
L'idée présentée ici exploite le fait que le champ scalaire fournit beaucoup d'informations sur la

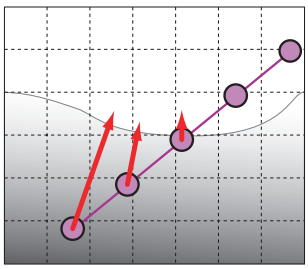


FIG. 1.99: Illustration du principe de retour d'effort à partir d'un champ potentiel scalaire (schéma d'après [AS96])

surface affichée. En effet, en un point donné, la valeur du champ indique directement si ce point est à l'intérieur ou à l'extérieur de la surface. Le gradient donne une indication sur la direction de la normale au voisinage de la surface et coïncide avec la normale sur la surface. Le potentiel scalaire peut encore être exploité pour déduire la pénétration du point, en effet sa valeur est liée à l'éloignement de la surface (voir Figure 1.99). Ils parviennent ainsi à mettre en place un dispositif de retours d'effort ponctuel permettant d'explorer des données volumiques.

2.5 Utilisation d'un gant de données

Geometric Manipulation of Tensor Product Surfaces

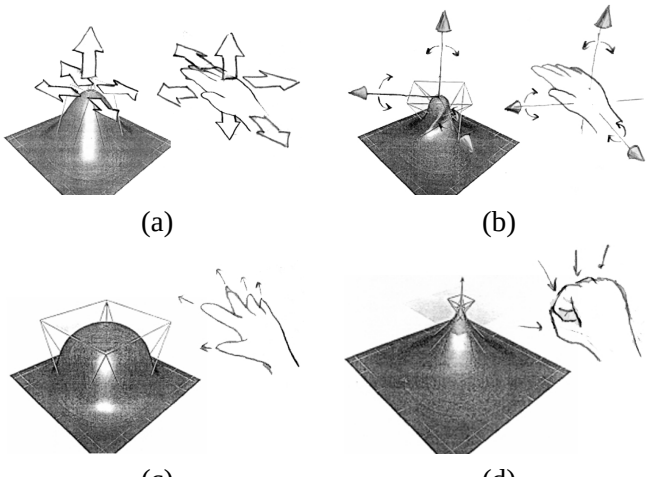


FIG. 1.100: Illustration des concepts de manipulation de surface : (a) positionnement du point sélectionné de la surface par translation de la main, (b) spécification de ses tangentes, et de la torsion (c) par rotation de la main. (d) et (e) contrôle de la *tension* de la surface (norme des tangentes) à travers la taille du carré fixée par les mouvements des doigts.

Dans la partie consacrée aux travaux futurs de [Fow92] (que nous avons abordé en 1.2), B. Fowler développait quelques idées pour spécifier ses contraintes géométriques à l'aide de posture de la main. Il envisageait d'utiliser un gant de type VPL-Dataglove pour les mesurer :

- le positionnement du point à modifier se fait par **translation** de la main. Une translation dans le plan horizontal équivalant à un déplacement du point sélectionné dans le domaine de paramètres (u, v) de la surface (figure 1.100.a).
- la **rotation** de la main sert à spécifier les positions des tangentes et à simuler une torsion de la surface (figure 1.100.b).
- la **fermeture** de la main sert à contrôler l'amplitude des tangentes, simulant une *tension* plus ou moins importante de la surface (figure 1.100.c et 1.100.d).

Surface Drawing

Une autre utilisation de la main a été proposée plus récemment par P. Schkolne et P. Schröder dans [SS99, SS01]. Le système utilise une interaction à deux mains. Une main sert avec un stylet à déplacer l'objet pointé. L'autre main sert à constuire des surfaces. L'article détaille les techniques

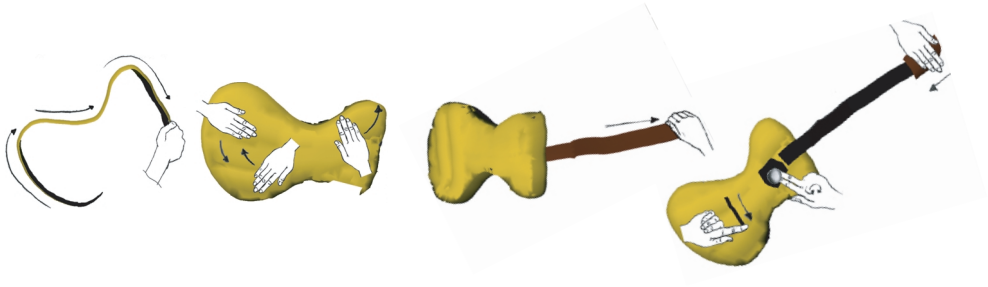


FIG. 1.101: Illustration du principe de construction de surfaces (images extraites de [SS99]).

utilisées pour construire interactivement un modèle polygonal suivant les gestes de la main. Trois modes sont accessibles selon la posture de la main pour dessiner, effacer, ou ajouter des détails fins. L'activation d'un mode dépend de la position du pouce (voir Figure 1.102). En mode dessin, la surface

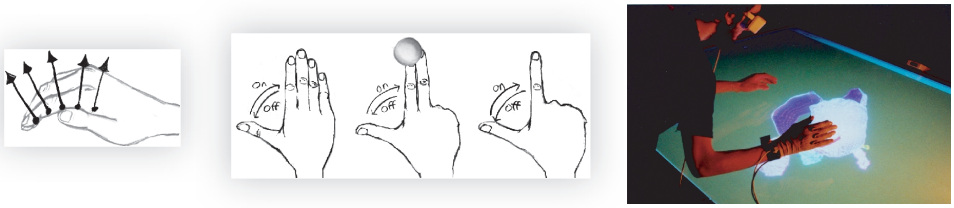


FIG. 1.102: (a) illustration des points échantillonnés pour construire la surface ; (b) conventions utilisées pour les différents modes dessiner, gommer, et ajouter des détails fins, chaque mode étant activé / désactivé par la position du pouce ; (c) exemple d'interaction avec un *Responsive Workbench* (images extraites de [SS99]).

est générée en suivant la position de quelques points de la main. On voit un exemple de session pour dessiner une guitare dans le Figure 1.102.

2.6 Utilisation d'un visiocasque

L'intérêt du visiocasque est qu'il permet de présenter une scène virtuelle immergeant complètement l'utilisateur, et masquant par là même les éventuels dispositifs utilisés pour lui restituer des sensations, comme par exemple un dispositif de retour d'effort.

3DM : A Three Dimensional Modeler Using a Head-Mounted Display

J. Butterworth et al. utilisent dans [BDHO92] un visiocasque non-transparent pour immerger complètement l'utilisateur dans l'environnement du modèle. Le but étant de l'aider à la compréhension des relations spatiales entre le modèle et le curseur dont il dispose.
Le curseur est contrôlé via une *souris-6D* à deux boutons (développée à Chapel-Hill, University of North Carolina) qui n'est pas exposée plus avant ici, l'article étant plus orienté vers la méthodologie que vers les technologies d'interaction.
Le modèle de surface utilisé est une hiérarchie de triangles. L'utilisateur se trouve sur un *tapis magique* qui matérialise la zone de validité des capteurs de position de la tête et de la souris 6D. Il dispose d'une *boîte à outils* sous la forme d'un menu flottant qu'il peut déplacer et actionner via un curseur lié à la souris. La forme de ce curseur dépend de l'action, de l'outil ou du mode sélectionné.
Pour observer le modèle, il peut se déplacer physiquement (marcher quelques pas sur le *tapis magique*), ou bien si le modèle est trop grand, il peut *voler* autour, le déplacer (*grab*) ou encore modifier son échelle.
Les outils permettent de créer une surface triangle par triangle, ou par extrusion. L'utilisateur peut grouper, déplacer, étirer un ensemble de triangles ou des sommets. Des opérations classiques comme couper, copier, coller, détruire et une pile de défaire / refaire sont également proposés.
Les auteurs concluent en essayant de dégager les forces et faiblesses de leur système. Les principaux apports sont une meilleure compréhension des relations spatiales (placement de primitives) ainsi qu'une maîtrise accrue et plus facile des outils classiques de modélisation comme l'extrusion.
En revanche, le manque de contraintes pour le placement à *main levée* avec la souris 6D entraîne une mauvaise précision, par exemple, pour créer deux polygones parallèles.

La fatigue causée par la station debout, le positionnement à bras levé du pointeur, ou l'utilisation immersive du visiocasque ne sont pas abordées.

Force and Tactile Feedback in a Virtual Environment

(a) (b) (c)

FIG. 1.103: Le cockpit virtuel de CGSD : (a) schéma de principe, (b) la prototype réalisé, (c) détail d'un panneau de commande (images extraites de `www.cgsd.com`).

R. Latham s'intéresse dans [Lat96] à la simulation d'un cockpit. L'idée est d'utiliser des instances réelles des contrôles existants dans l'environnement à simuler (boutons poussoirs, interrupteurs, ...) et de les placer sous la main de l'utilisateur dans l'état attendu (figure 1.103.a). Le prototype réalisé est constitué de (figure 1.103.b) :

– un visiocasque qui immerge totalement l'utilisateur et affiche le cockpit avec ses divers indicateurs et commandes, la vue extérieure au cockpit et une représentation de la main de l'utilisateur, dont la position et la posture sont mesurées grâce à un gant.

– un panneau capable de positionner et configurer les contrôles réels pour les amener sous la main de l'utilisateur dans un état cohérent avec l'affichage. Ces contrôles sont de type interrupteurs, boutons poussoirs, boutons rotatifs, ... (figure 1.103.c)
– trois ordinateurs : un pour le suivi des positions du visiocasque et du gant, le second pour la génération des images, et le dernier pour asservir le panneau.

Les auteurs semblent satisfaits des résultats obtenus : les contrôles sont simulés efficacement aussi bien pour la sensation tactile que pour le retour d'effort. Ils jugent cette approche prometteuse et extensible à d'autres domaines comme la validation de tableaux de bords (automobiles ou autre) ou tout panneau de contrôle en général.

Cette approche qui, à priori semble complètement inutilisable, dans un cadre de sculpture virtuelle pourrait éventuellement être exploitée en présentant sous la main de l'utilisateur une interface de retour tactiles comme vu dans la section 2.1.

2.7 Utilisation de visiocasque semi-transparent

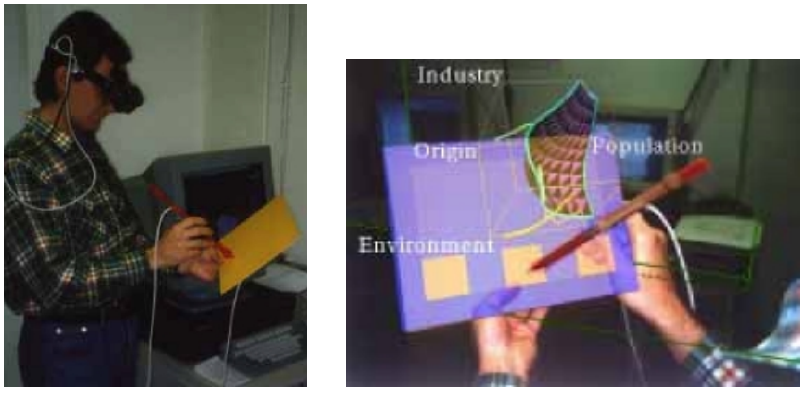
Comme on l'a déjà évoqué, le fait d'enfermer complètement l'utilisateur dans un environnement virtuel peut être gênant, ne serait-ce que si le téléphone sonne, pour reprendre le propos de E. Lantz dans [Lan97], mais aussi pour des raisons de fatigue / confort ou des besoins de travail collaboratif.

The Personnal Interaction Pannel : a Two-Handed Interface for Augmented Reality

Le concept de *PIP*, signifiant *Personnal Interaction Pannel*, introduit dans [SG97] reprend le concept de *bloc-notes* avec stylo pour son système d'interaction à deux mains et propose d'utiliser une tablette dans une main et un stylet dans l'autre.
Plusieurs configurations sont exposées, faisant appel à une tablette plus ou moins évoluée :

– utilisation d'une tablette affichant une image stéréoscopique (i.e. couplée à des lunettes obturatrices à cristaux liquides) à la manière d'un mini-Responsive Workbench, et dont l'écran est sensible à la pression.
– utilisation d'une tablette sensible à la pression couplée à un visiocasque transparent.
– la configuration *dumb panel*, utilisant une tablette *inerte* et *vide* servant uniquement au retour tactile pour l'utilisateur, et un visiocasque transparent qui sert aussi à afficher les informations sur la tablette.

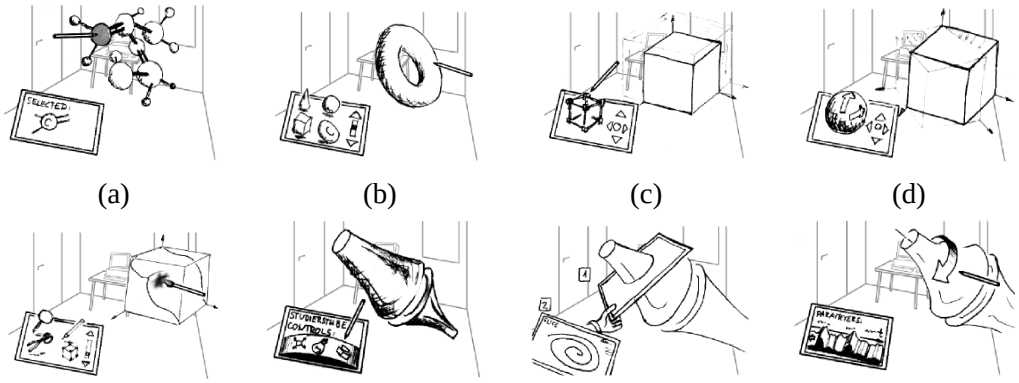
La configuration exploitée dans [SG97] est celle du *dumb panel* (figure 1.104.a).



(a) (b)

FIG. 1.104: (a) Exemple en fonctionnement d'une réalisation matérielle du PIP. (b) exemple de vue obtenue par superposition d'images synthétiques grâce au visiocasque transparent (images extraites de [SG97]).

La tablette est utilisée comme support pour une boîte à outils ou panneau de contrôle (figures 1.104.b et 1.105). Sa manipulation se révèle plus intuitive que les menus flottants de [BDHO92].
Elle peut également être utilisée dans le cadre de visualisation de données (figures 1.104.a et 1.105.h) et aider à spécifier des données géométriques, comme des plans de coupe (figure 1.105.g).
La conclusion de l'article insiste sur la gêne introduite par les câbles des capteurs et la faible qualité



(a) (b) (c) (d)

(e) (f) (g) (h)

FIG. 1.105: Exemples d'utilisation possible du PIP : (a) sélection directe de parties d'un modèle. (b) sélection dans une palette d'objets préfabriqués et insertion dans la scène par *glisser-déposer*. En plus des manipulations directes, (c) adaptation d'échelle et (d) rotation plus précise via des *widgets 2D*. (e) des outils plus généraux comme une loupe, un ciseaux (couper, coller) ou un pinceau. Parmi les autres utilisations exposées, citons (f) le réglage des paramètres de l'application sans quitter l'environnement. Aussi, dans le cadre de la visualisation scientifique, on peut utiliser la tablette pour spécifier des données géométriques, comme des plans de coupe (g) ou mesurer des paramètres du modèle (f) (images extraites de [SG97]).

des images obtenues avec le visiocasque. Les auteurs révèlent toutefois être encouragés dans cette voie par les premiers utilisateurs qui ont appréhendé quasiment seuls, et trouvé très intuitive, l'utilisation du PIP. Ils n'ont pas reporté de fatigue, concernant aussi bien le visiocasque, que le poids de la tablette ou le maintien tu stylet sans appui.

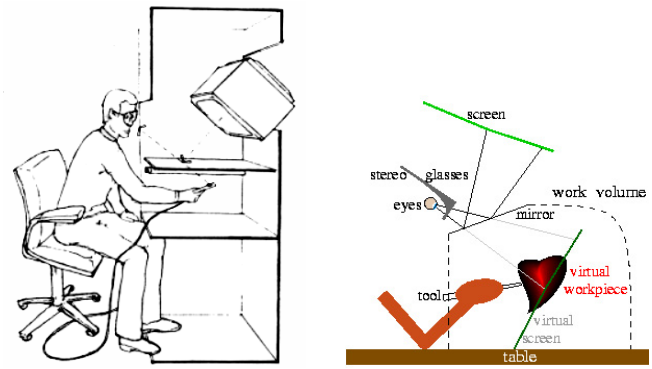
Une extension multi-utilisateurs du PIP, appelée *Studierstube* a également été présentée par la même équipe.
Ce type d'interface est évidemment très séduisant dans un contexte de sculpture virutelle, puisqu'il contribue à immerger la sculpture dans l'environnement de l'utilisateur, et fournit un moyen simple et intuitif pour disposer d'une boîte à outil, sans avoir recours à des postures codifiées par exemple.

2.8 Utilisation d'un Virtual Workbench

Ce dispositif, à ne pas confondre avec le Responsive Workbench ou Immersive Workbench évoqué dans la section 2.1 permet comme les visiocasques semi-transparents de fusionner l'espace de travail et l'espace de visualisation.

Dextrous Virtual Work

T. Poston and L. Serra présentent dans [PS96] un système appelé *Virtual Workbench* qui consiste à projeter l'image d'un écran sur un miroir, pour que l'utilisateur, muni de lunettes obturantes à cristaux liquides, puisse glisser ses mains dessous et ainsi avoir la sensation de voir un objet à la position exacte où se trouvent ses mains : dans le volume de travail situé entre le miroir et la table (voir figure 1.106.b). Aidé par un retour d'effort, dont la mécanique est alors totalement cachée, l'utilisateur peut



(a) (b)

FIG. 1.106: (a) description de l'environnement utilisé en 1983 dans [Sch83] (image extraite de [Sch83]). (b) description du *Virtual Workbench* de T. Poston and L. Serra [PS96] (image extraite de [PS96]).

même avoir la sensation de toucher cet objet.
Un dispositif analogue avait été proposé par C. Schmandt en 1983 [Sch83] utilisant des lunettes obturantes à cristaux liquides, un capteur magnétique de position et orientation, et un miroir **semi-transparent** laissant l'utilisateur voir sa main (figure 1.106.a). Les problèmes d'occultation posés lorsque la main est physiquement sous un objet mais demeure tout de même visible s'avèrent très perturbants. Egalement, les réglages de contraste entre les images générées et la main sont problématiques. Les applications envisagées dans [Sch83] étant plutôt orientées vers le *design* de circuits VLSI, les limitations matérielles de l'époque (notamment, la mauvaise précision du capteur magnétique sous l'écran cathodique) ont nuancé le succès de cette approche.

Le but poursuivi ici par les auteurs était de pouvoir assurer un grand confort d'utilisation pour à la fois permettre une utilisation **prolongée** et **précise**. Par rapport à un visiocasque, ce dispositif offre une grande résolution pour un coût raisonnable, ainsi qu'une grande stabilité, car l'affichage n'est pas asservi avec la position de la tête de l'utilisateur. Le fait de placer le modèle dans l'environnement de l'utilisateur et pas l'inverse, comme le font les procédés immersifs, s'avère moins déstabilisant. Tout cela combiné à la possibilité d'utilisation assis, avec les bras en appui sur une table contribue au **confort** du système.

La précision est également fortement dépendante des capteurs utilisés. Les auteurs signalent avoir essayé une 6D-mouse de Logitech et un *Bird* d'Ascension Technologies qui se sont révélés trop instables. Un capteur Fastrak de Polhemus, ou un *Probe* d'Immersion (bras articulé ?) se sont révélés moins sensibles aux perturbations magnétiques dues à l'écran ou aux réflections ultrasonnores du miroir.
Plusieurs applications médicales de diagnostique et de planification d'opérations du cœur et du cerveau ont été entamées. L'interface proposée à l'utilisateur dans ces applications fait appel à une barre d'outils *Tool Rack* statique au fond de l'écran, permettant de choisir des opérations classiques (loupe, copier, couper, coller, entrées / sorties, ...) ou des outils spécifiques (scalpel, pied-à-coulisses, marteau, stylo, ...)

Dans le cadre d'une utilisation prolongée, ou la station debout deviendrait problématique, ce dispositif semble bien adapté à un travail précis sur des formes dont les dimensions sont de l'ordre du volume de travail.
L'utilisation d'un bras articulé à retours d'effort fournit de plus mécaniquement une position, ce qui évite les problèmes écoqués d'instabilités des capteurs magnétiques.

Ce qui est très intéressant ici c'est la fusion du volume de travail et du volume d'affichage de l'objet virtuel, sans imposer le port d'un casque ; la sensation de contact direct avec la surface est renforcée par cette superposition.

Conclusion

Après ces explorations des techniques de modélisation et d'interaction, on peut essayer de dresser un bilan sur les points importants vis-à-vis de la sculpture virtuelle.

Dans l'idéal, on devrait pouvoir ne pas se rendre compte du côté virtuel pour tout ce qui concerne les aspects manipulation, vision, perception et juste bénéficier des avantages de cette virtualité. Par exemple le contrôle complet sur les matériaux utilisés (pas de contrainte de cassure, séchage, pesanteur, équilibre, ...), possibilité de revenir en arrière sur les modifications, confort d'utilisation, contrôle des dimensions, ...

Comme nous ne sommes pas dans un monde idéal, cette première partie nous a servi, à partir de ce qui existait, à identifier les contraintes et à faire des choix quant aux aspects à abandonner ou privilégier.

Par exemple, nous avons décidé très tôt de ne pas simuler physiquement, c'est-à-dire avec des paramètres mesurés, des lois de comportements et des intégrations dans le temps, les matériaux utilisés. Evidemment pour des considérations de complexité et de ressources machine, mais aussi parce qu'éventuellement, cette simulation peut occasionner des altérations non souhaitées de la surface et provoquer une gêne plutôt qu'une aide dans sa construction.

Il était clair par contre dès le départ, que l'objet central d'intérêt devait rester la surface sculptée. En particulier, toute structure de contrôle interne à une représentation donnée apparaissant dans le processus constituait en ce sens, au mieux, une distraction, au pire, une contrainte. Dans ce contexte, la manipulation directe de la surface nous est apparue essentielle.

Dans la première partie nous avons vu que malgré leur popularité et la richesse de leurs techniques d'édition, les surfaces de subdivision, que l'on a considérées ici comme unificatrices des représentations polygonales et splines, ne parvenaient pas à masquer cette structure de contrôle interne. Cette structure, i.e. le polygone de contrôle, impose des restrictions sur l'édition : il faut planifier sa construction pour profiter des contrôles sur la surface aux endroits souhaités. Egalement, ce polygone doit avoir la même topologie que la surface finale, et il faut donc le manipuler explicitement pour gérer ces changements. Cette limitation peut aussi conduire à une structure de contrôle aussi complexe que la surface elle-même dans le cas de surfaces compliquées.

Ainsi, parmis les représentations et techniques d'éditions que nous avons pu recenser, seules les surfaces implicites nous sont apparues satisfaisantes pour parvenir à masquer toute structure de contrôle interne. En particulier les représentations discrètes des surfaces implicites présentent l'avantage de ne pas tenir compte du nombre et de la complexité des opérations d'éditions tout au long du processus de construction.

La seconde partie nous a montré les possibilités quant aux interfaces de restitution tant visuelle, que haptique. Nous avons abordé quelques exemples de systèmes potentiellement prometteurs pour soit placer l'utilisateur dans l'environnement de la sculpture, soit l'inverse. Au vu des avancées technologiques et des ressources machine, la solution le plus prometteuse au début de ces travaux nous a semblé être un périphérique de type bras à retours d'effort.

CHAPITRE 2

Premier prototype

Ce chapitre reprend l'article présenté à Implicit Surfaces 99 (septembre 99) et retenu pour publication dans le journal Visual Computer. Des versions intermédiaires ont été présentées également sous forme de short paper à Eurographics (septembre 99) et en français à l'AFIG (novembre 99). La version la plus complète étant celle parue dans le numéro 8 volume 16 du journal Visual Computer en décembre 2000.

Abstract

We present a *sculpture metaphor* for rapid shape-prototyping. The sculpted shape is the iso-surface of a scalar field spatially sampled. The user can deposit material wherever he desires in space and then iteratively refine it using a tool to add, remove, paint or smooth some material. We allow the use of freeform tools that can be designed inside the application. We also propose a technique to mimic local deformations that allow to use the tool as a stamp to make some imprints on an existing shape.
We focus on the rendering quality too, exploiting lighting variations, and environment textures that simulate high quality highlights on the surface. Both greatly enhance the shape estimation, which is a crucial step during this iterative design process, in our opinion. The use of stereo also greatly eases the understanding of spatial relationships.
Our current implementation is based on GLUT and allows the application to run both on UNIX-based systems, such as IRIX and Linux, and on Windows systems. We obtain interactive response times, strongly related to the size of the tool. The performances issues and limitations are discussed.

Key words : sculpture metaphor, discrete implicit surfaces, local deformations, balanced binary trees, hash tables.

2 Introduction

The general context of our work is 3*D* freeform design. We deliberately chose a *sculpture metaphor* based on iso-surfaces over a sampled scalar-field. To present our choice and place our approach in the context of freeform modeling, we propose a simplified classification based on some major/most recent contributions that came to our knowledge.

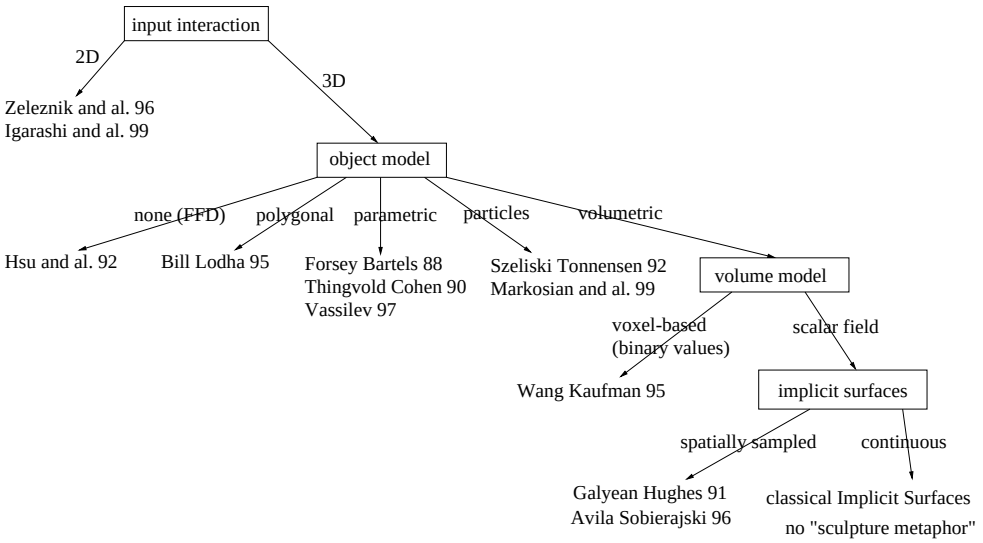


FIG. 2.1: Proposed simplified classification of approaches to rapid prototyping of 3*D*-shapes

The first distinction we make concerns the input device used. One possible approach to designing 3*D* shapes is to use 2*D* input devices, such as a classical mouse or a pen to draw strokes on a screen, and then deduce plausible 3*D* shapes fitting these silhouettes strokes. Some interesting approaches have been conducted such as the SKETCH system by [ZHH96] that enables rapid shape construction using simple primitives (boxes, spheres, cones) that are combined with CSG-like operations and positioned with gestures. [IMT99] extended this approach with the Teddy system which uses freeform strokes to deduce 3*D* polygonal model and then simple gestures to extrude, smooth, cut or deform (bend) it. This approach is very appealing both because of its close relation to the pen and paper interaction, which is natural when roughly sketching ideas in early stages of design, and because of its simplicity in software interface (no sliders, buttons, confusing options) and in hardware interface (simply a pen).
Nevertheless, we preferred 3*D* input devices (also called *direct* or 6*D* input devices), as we believe it allows a better control over the 3*D* shape. When considering more complicated devices, such as the Phantom force-feedback articulated arm, the device also allows to *feel* the surface being modeled, which greatly enhances the surface perception, and consequently its valuation.

The second distinction deals with the underlying model description, which also strongly conditions the range of achievable objects. We only consider here approaches based on direct surface interaction. [HHK92] used Free Form Deformations, which are powerful to deform any kind of object. [CLE98] mixed these deformations with a force feedback device, which leaded to a convincing clay modeling interaction. However, the serious limitation of this space deformation technique is that the user can not modify the topology of the deformed object. [BL95] proposed a polygonal sculpture system, where the user is able to alter the topology, but he has to specify the new connectivity at a triangle level. Moreover, when the user deforms the shape, detecting and handling its self-collisions becomes very intricate. [FB88] initiated another interesting approach with their Hierarchical B-Splines, that allowed to add details via *overlays* locally refining the shape. Intuitive spline manipulation may be extended with physical simulation processes such as proposed by [TC90] and [Vas97]. But here again,

topology changes, especially when self-intersections occur, are very difficult to handle (and are not handled, to our knowledge).
Another interesting class of approaches, that we call *particle based*, was initiated by [ST92]. [WW94] extended this, notably by maintaining a triangulation over the particles. The user can interact with the surface and make some holes in it or specify connections via control curves drawn onto it, thus having a good control over its topology. But here again, self-intersections are very complex to detect, and *ill-formed* surfaces, such as the Klein-mug shown in the article can be built.

Volumetric models enable one to easily represent 3D shapes of any topology and prevent the construction of self-intersecting surfaces. We put in this category *classical* implicit surfaces (a good description may be found in the book from [BBB+97]). However, building a sculpting metaphor from classical implicit surfaces seems difficult. Indeed, these surfaces are usually defined as the blending of elementary potential fields generated by individual primitives. The number of these primitives greatly affects the field evaluation cost. Since iterative shape refinement is a key aspect of sculpting, a classical implicit representation would lead to a complexity explosion due to the increasing number of primitives.

In this context a representation of the field function as values stored in a grid seems much more appropriate. Then, whatever the number of editing operations, trilinear interpolation always evaluates the field value in constant time. We adopted here this representation, which [GH91, WK95, AS96] already studied.

We extend the tools shapes to freeform models that can be designed inside the application and propose a new tool action which allows to use the tool as a stamp printing its shape on the virtual clay. We also pay attention to the rendering quality which greatly enhances the perception of the 3D shape as we think this is a crucial feature to estimate each modification along the design process.

We firstly describe our implementation through a couple of *how does it work* sections : the first one is dedicated to the potential field storage and update, and the second one is dedicated to the tools shapes and actions. Finally, we detail our visual feedback, estimate our system performances on sample cases, and discuss future works.

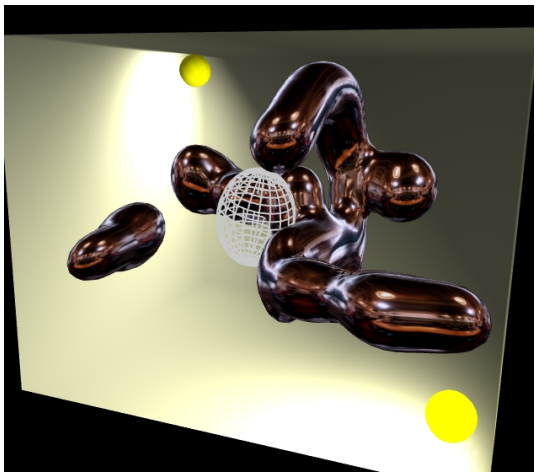


FIG. 2.2: Sample snapshot of our sculpting application. The object being modeled is environment mapped. The tool is displayed in wireframe mode. The two yellow spheres represent the lights and a box representing the workspace is displayed.

3 Discrete potential field storage

We considered the physical size of the sampling grid directly implemented as an array, such as in [GH91, WK95, AS96], as a limitation : not only because it encloses the model and limits its

extension, but also because it wastes memory storing irrelevant sample points where no potential field is defined.
We preferred to use a dynamical structure to store only the *relevant* sampling points. To that end, we first decided to use balanced binary search trees, which are less efficient than hash-tables in best case, but have better operation time in worst case configuration. When we lately decided to implement hash-table based version to compare with, using a very classical hashing function, such as the one that [Blo94] proposed, we found the transition very easy since the encapsulating data structures remain unchanged. In the following, we firstly describe these data structures, trying to give our motivation for each structure. In section 3.2 we explain how these structures interact when we edit the field.

3.1 Data structures : static description

We call the regularly spaced points that sample the potential field `Corners`. Each of them stores a potential field value, a color and some cached data, such as the field gradient, and the point location (to avoid its recomputation from the virtual grid indices and sampling steps). We call the structure used to store/retrieve/update them a `CornersTree`.
In our **balanced binary tree** implementation, each `Corner` possesses a key, so that it can be compared to the others and inserted in the tree. The key we use is simply made up of the indices (i, j, k) of the `Corner` in the virtual grid implicitly defined by the regular space sampling. Two keys (i_1, j_1, k_1) and (i_2, j_2, k_2) are compared in lexicographical order, which means that we first compare i_1 and i_2. If they are equal, we then compare j_1 and j_2, and finally if j_1 equals j_2, we compare k_1 and k_2.
In our **hash-table implementation**, we directly used the SGI's STL hash-table template. We only had to provide a hash-function, inspired from the one proposed by [Blo94] : the key is a 15 bits integer, concatenation of the 5 least significant bits of i, j and k.
Each `Corner` having a value between an arbitrary `minVal` and `maxVal` (arbitrarily set to 0 and 3 in our case) is stored in the `CornersTree`. A corner whose value becomes less than `minVal` is removed from the structure and deleted. Values above `maxVal` are clamped to `maxVal`. When requesting the value of a `Corner` not defined, the returned value is `minVal`.

This regular space sampling divides the space in cubical elements we name `Cubes`. In a similar manner, each `Cube` having at least one `Corner` defined is stored in a `CubesTree`. The key associated with each `Cube` is a triplet (i, j, k) that corresponds to the smallest `Corner`-key of the eight `Corners` defining it. A `Cube` is made up of :

- eight pointers to its `Corners`, one of which is non-null, at least.
- an index deduced from the value of its eight `Corners` relative to the iso-value, which encodes the `Cube`/iso-surface intersection configuration. This is a classical decomposition step from the Marching-Cubes algorithm (furtherly presented by [Blo87, LC87, Blo88])
- twelve pointers to edges.

The `Cubes` that intersect the iso-surface (depending on their index value) are also inserted into another structure (tree or hash-table) which we call `crossList`. Figure 2.3.a shows all the `Cubes` contained in `CubesTree`; Figure 2.3.b shows the cubes crossing the surface (stored in `crossList`); Figures 2.3.c and 2.3.d shows the corresponding iso-surface with different rendering modes.

An `Edge` is created only to compute and store an intersection with the iso-surface. `Edges` are stored in an `EdgeTree`, which underlying structure is a balanced binary tree or a hash-table. An `Edge` key is the concatenation of the two `Corners` keys forming the edge (with the smallest `Corner` first to ensure uniqueness).

3.2 Applying a tool : data structures update

When a tool is applied, we have to flush its modification in the `CornersTree`. To this end, we first compute the axis-aligned (grid-aligned) bounding box surrounding the local tool bounding box.

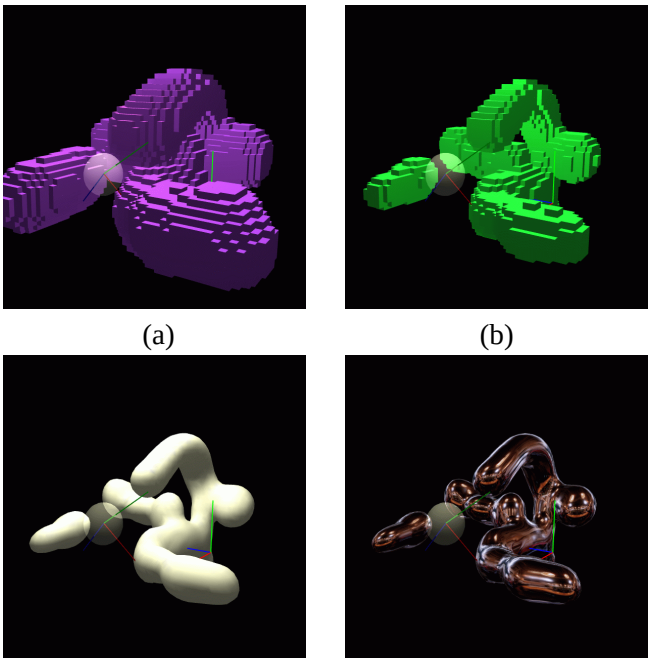


FIG. 2.3: Visualization of some data structures elements : (a) shows all the cubes having at least one corner defined. (b) shows all the cubes that cross the iso-surface. (c) shows the extracted iso-surface. (d) uses environment texture to improve the shape perception when the user moves around.

Then, we have to walk through this box by :

1. transforming from world to local (tool) coordinate only the two extremum points of the box (P_{min} and P_{max}) and the three displacement vectors (that move from one `Corner` to the next in each axis direction).
2. starting from the P_{min} point, we can reach the next point simply by adding its displacement vector to its current location, and similarly adding its counterpart displacement vector to its counterpart location in local (tool) frame coordinate (see Figure 2.4). Note that any scaling can be applied to the tool by applying the inverse scaling to the local location we just obtained.

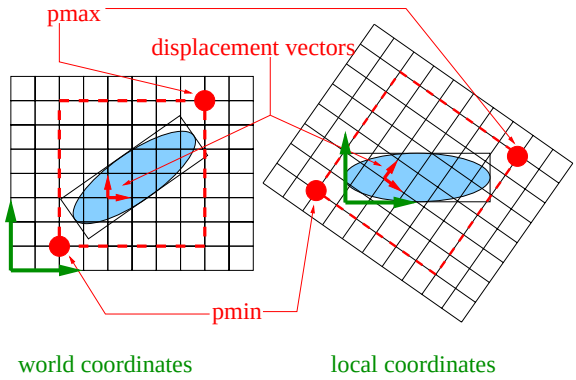
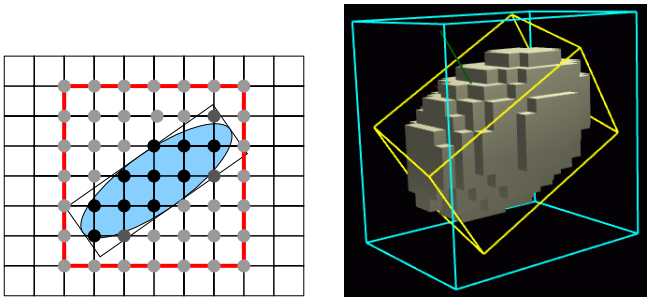


FIG. 2.4: Parallel bounding box walkthrough is conducted both in world and local (tool) coordinates. Only the two points P_{min} and P_{max}, and the three axis-aligned displacement vectors are transformed from world to local coordinates.

For each `Corner` examined during this walkthrough, we distinguish three cases (see Figure 2.5.a) :

1. the `Corner` is in the world bounding box, but outside of the local bounding box. It can be very quickly rejected, since the bounding box containment is a very rapid test in the local frame coordinate. We call these `Corners` the **visited** `Corners`. They appear light grey in Figure 2.5.a.
2. the `Corner` is inside the local bounding box, but outside the tool's influence (i.e. the tool's potential field has a null contribution at this point). To identify this case, we must compute the tool's potential value for that point ; we call them the **computed** `Corners` (meaning that we computed the tool's potential field, but finally the `Corner` wasn't modified). They appear in grey on Figure 2.5.
3. the last category concerns the `Corners` whose value was effectively modified. We call them the **dirtied** `Corners` because they have to be updated (cleaned). They appear in black on Figure 2.5.a.

All these **dirtied** `Corners` are inserted into a temporary tree called `modified`.



(a) (b)

FIG. 2.5: Applying the tool. (a) represents in 2*D* the virtual grid. Light-grey points represent the `Corners` **visited**. Grey points represents the `Corners` **computed**. Dark-grey points represent the `corners` **dirtied**. (b) shows the axis-aligned (blue) and oriented (yellow) tool bounding boxes. The cubes displayed are the `Cubes` which possess at least one `Corner` that has non-null contribution from the tool.

Each time a redisplay is needed, we successively extract (pop) every `Corner` from the `modified` tree. For each `Corner`, we then have to update the eight `Cubes` that share it. To avoid multiple `Cube` examinations, we used either a timestamp-mechanism or another temporary tree to store only once each *dirtied* `Cube`.
To examine a `Cube` we compute the index (i.e. a bitmask deduced from the `Corners` value relative to iso). If the `Cube` doesn't cross the iso-surface, we're finished with it. If it crosses the iso-surface, its index corresponds to a surface crossing configuration in a precomputed table (classical part of the Marching Cubes process). This configuration tells us which `Edges` of the `Cube` are intersected. The corresponding `Edges` are then updated.

When an `Edge` is updated (or created), the field gradients of its two extremity `Corners` are first (re-)computed (with a central difference scheme in our current implementation). Then, the intersection point is obtained by linearly interpolating each `Corner` attribute (such as the location, gradient and color) weighted by the corresponding potential field value stored. The interpolated gradient serves as surface normal (and is consequently sent to the graphic hardware).

3.3 Undo/redo handling

One key feature to encourage creative explorations is to allow multiple successive tries : the user can experiment whatever he desires without any consequence because he can always return to an

earlier configuration.
We achieve the undo/redo process via temporary *undo-files* : each time a tool is applied, we dump all the modified `Corners` into a new *undo-file*.
In our current implementation, dumping a `Corner` corresponds to :

1. writing its indices in the virtual grid (i.e. the triplet (i, j, k) relative to its current origin and step size.
2. writing its previous value and attributes (color only in our case, the other attributes such as the location and gradient are simply caches, and can be computed).
3. writing its new value and color after modification.

For example, the tool in Figure 2.5.b corresponds to 521 `Corners` dirtied (796 `Cubes` are displayed) ; the undo-file size is around $15kB$ (14588 bytes, exactly). We arbitrarily limit the number of undo-files to 200. When more than 200 modifications are conducted, we cycle through the existing undo-files (restarting from the beginning). As the filesystems' caching is sufficiently efficient to achieve these dumps at interactive rates (both under IRIX6 and WindowsNT4), the real limitation to this undo/redo process is only disk space.

4 Sculpting tools

A tool is defined by :
– a **potential field**, that defines what we called the *tool contribution* in space. The tool bounding box is in fact the bounding box of this field. This potential field also indirectly (implicitly !) defines the tool's shape, which corresponds to an iso-value of the field.
– an **action**, that defines the way the tool's potential field is combined with the (possibly) existent object's potential field.

In the next section we explain what are the possible shapes/potential fields for our tools. Then, we go through the possible tool's actions : after describing how we implemented the classical tools actions, we present our method to achieve local deformations to make some imprints on an object.

4.1 Tool shape

Previous approaches were all using binary sampled volumes as primary tool potential : having a 1 value inside, and 0 value outside for example. Thus they always had to filter this potential (via expensive convolutions) in order to prevent aliasing. This notably restricted the tool shapes used by [GH91] and [AS96] : each orientation change of a non-isotropic tool would need a new filtering operation, that could not be achieved at interactive rates. So they restricted themselves to sphere-shaped tools that were filtered in a preprocess stage. [WK95] are less strict with the antialiasing pass, and allow any orientation/scaling of the 20^3 grids storing the prefiltered tool potential samples.

We are even less strict here, as we directly use a continuous tool potential field without any filtering pass. We assume that the sampling rate is sufficiently small to avoid noise artifacts. As this sampling rate is constant and regular, this comes to using tools not too small relatively to the sampling grid spacing. We let this to the user's responsibility or choice, as there is also a trade-off between the tool size and the field update rate, as we will see in section 6.

We use the continuous *Wyvill* potential such as proposed by [WMW86], or a box-shaped function (see Figure 2.6) to generate an isotropic (spherical) field around the tool center. We obtain general ellipsoids by scaling the tool along its three axes. In this particular ellipsoidal case, the shape displayed to represent the tool in the workspace is the limit of the influence region.

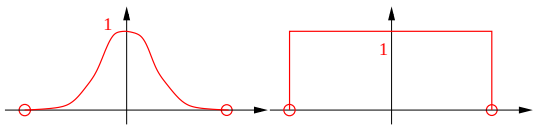


FIG. 2.6: Continuous and box-shaped field functions used for the ellipsoidal tool

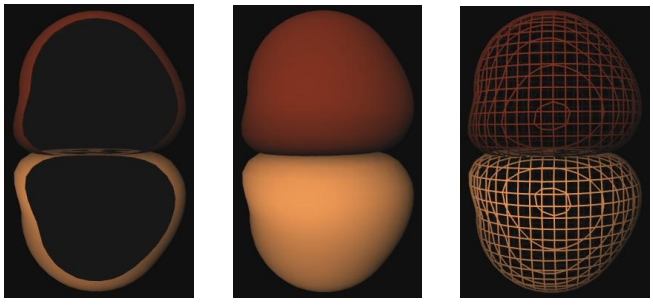
We also propose freeform tool shapes that can be generated inside our application. To achieve this, we duplicate the structures storing the corresponding discrete potential field. The shape displayed corresponds to the iso-surface, which is the same as the one visualized during its design process. The tool can also be scaled along the three axes of its local frame coordinate.
Since applying the tool requires the evaluation of its potential field between its samples' location, we *reconstruct* a continuous potential field by trilinear interpolation inside the `Cube` the point falls into : as the `Cube` stores pointers to its `Corners`, we save time to retrieve their value (as we avoid the whole `CornersTree` search for each of the eight values).

4.2 Classical tool actions

The classical tool's actions are, similar to the concepts presented by [GH91, AS96] :
– deposit material, which means that we **add** the tool's contribution to the (possibly) existing sample point (i.e. `Corner`).
– remove material, either smoothly (which means that we subtract the tool's contribution to the `Corners`) or not (which means that we delete all the `Corners` where the tool has a non-null contribution).
– paint material by changing the color attributes of the `Corners`. Again either smoothly or not, depending on if we take into account the tool's contribution respectively to the field value of the `Corner` being examined, or not.
– smooth the shape being modeled, which is indirectly conducted via the local field smoothing. This corresponds to a low-pass filtering operation conducted over the `Corners` covered by the tool.

4.3 Local deformation tool : use the tool as a stamp

Our aim here is to produce a visually convincing deformation while avoiding the computation cost and stability problems of a physical simulation of the material displacements. Our method is inspired from the [CD97, OC97] approach developed for classical (continuous) implicit surfaces, which consists in applying a negative field to compress the object in the area where another object penetrates it, while creating a bulge to imitate material displacement around this area, as illustrated in Figure 2.7.



(a) (b) (c)

FIG. 2.7: Images obtained with our implementation of the [OC97] method inside the iMAGIS-team implicit modeler : *Fabule* (further information about this modeler may be found in [RCG+97]).

We keep the same underlying ideas :

1. direct use of a bulging potential field to achieve *geometric deformations* without any physical simulation.

2. use the composite of the potential field of the colliding object (the tool in our case) and an *ad-hoc* deformation function to produce the bulging potential field.

But we use a slightly different deformation function, and as we use a discrete potential field, we are able to mimic plastic deformations, which were not practical in the original approach.

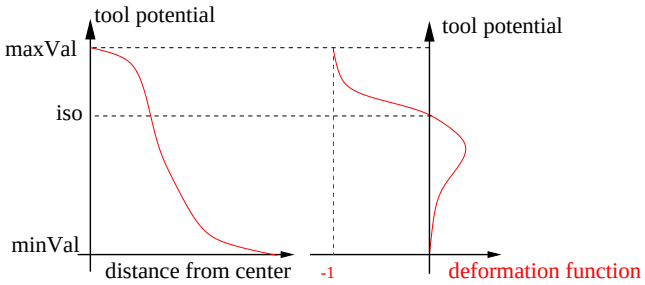


FIG. 2.8: Deformation function principle. On the left, we see the tool potential generated by a spherical tool : this potential varies with the distance from the center of the tool. On the right, we see how the deformation function is mapped onto the tool's potential. Composing the two function relates the deformation function to the distance from the center of the tool.

Our deformation function comes from simple intuitions : inside the tool, we should remove some material ; outside the tool, we should add some material in order to imitate the real material displacement that occurs when a real tool collides with a block of clay. As this inside/outside knowledge is already encoded in the potential field (value greater than iso meaning inside in our case ; the lower the potential value, the farther away from the iso-surface), we use the potential field of the tool as input parameter for our deformation function (see Figure 2.8).

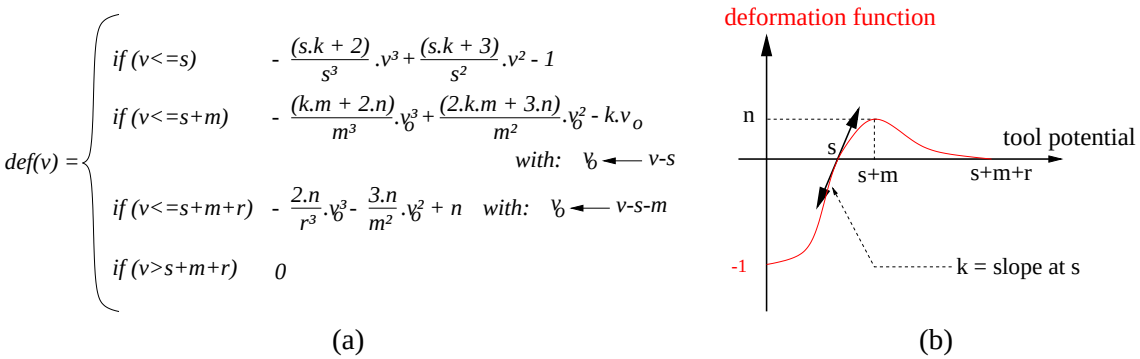
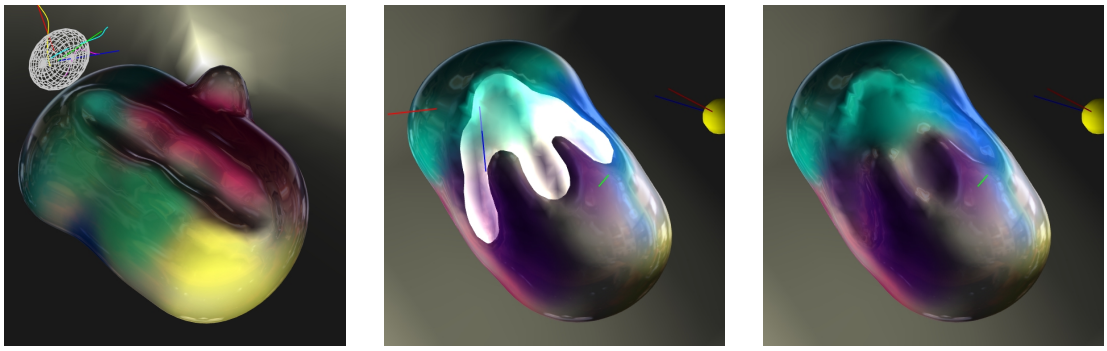


FIG. 2.9: Deformation function : (a) equation et (b) parameters.

The deformation function parameterization we are currently using appears in Figure 2.9. We have to make sure that the sum $iso + m + r$ doesn't bypass $maxVal - minVal$, otherwise the domain of definition of the deformation function becomes greater than the potential value variation : the deformation function would be truncated.

To sum up, we evaluate the contribution of a tool at a point P by first computing the value $v_{tool}(P)$ of the tool's potential field at P. The final tool's contribution is the composite of v_{tool} and def : $v_{deformTool} = def\,(v_{tool}(P))$. We can see some examples in Figure 2.10.



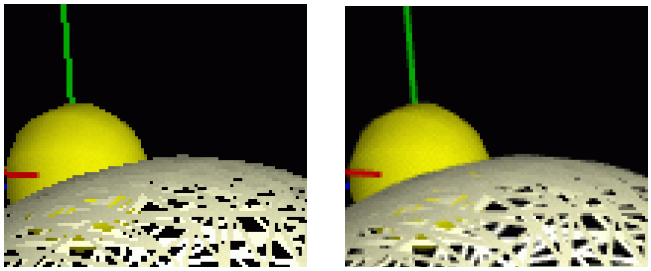
(a) (b) (c)

FIG. 2.10: (a) Imprint made by shifting an ellipsoidal tool onto an object. Using a tool designed inside the application as a stamp to make an imprint. (b) the tool is displayed in transparent mode. (c) the same view without the tool

5 Visual feedback

With this test platform, we soon understood how crucial the visual quality is for the user's comfort, but also for the tool's position perception, and for the object's shape estimation. In the following, we explain how we improved the rendering quality using simple OpenGL features.

We used the *Infinite Reality* graphic card ability to antialias OpenGL primitives at *no cost* thanks to its hardware support of the multisample extension (see Figure 2.11).
We also tried some stereo rendering using some *Stereographics* shutter-glasses (*Crystal Eyes* mo-



(a) (b)

FIG. 2.11: We see a close-up of a snapshot from our system. (a) without the multisample extension and (b) with.

del), and two types of HMDs (*i-Glasses* model from *virtual-IO* and *Proview60* from *Kaiser Electro-Optics*). Both are still in an early stage of development since we do not correctly handle the convergence/zero-parallax problem, and we do not track the head position. Even in this simple configuration, this proved very helpful for the tool placement in space.
Another feature which greatly enhances the shape perception is the use of environment textures that are *sphere-mapped* onto the object. We were first guided in that direction by some techniques using textures to simulate high quality highlights. This reveals particularly useful if the surface has degenerated triangles (see Figure 2.12), which is a typical drawback of the Marching Cubes algorithm. We used classical sphere-mapping with adjustable transparency to see the surface color under the texture layer. We also implemented simplified *ClearCoat*[1] like effect, i.e. simulating a paint layer. We simply used the transparency component of a texture and made it vary with the incidence angle between the viewer and the surface, which is related to the spheremap texture coordinate. [CON99] details some

[1] information concerning SGI's *ClearCoat* product may be found at `www.sgi.com/software/clearcoat`

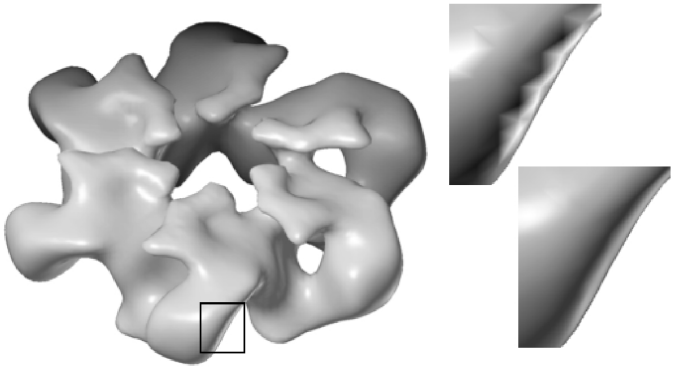


FIG. 2.12: Sample shading artifacts due to the object triangulation, and a proposed solution based on textures to render high quality highlights of infinitely distant light sources (image from [MBGN98], pp.273)

techniques to model more complex and physically accurate surface properties.

6 Performances and Results

We obtain interactive response times without the need of any dedicated/specific volume rendering hardware. At the expense of reduced performance and visual quality (no *multisampling* antialiasing, and slower frame rates), our application also runs on a standard PC using OpenGL under WindowsNT.

[GH91] used grids from 30^3 up to 60^3, while [AS96] reports the use of grids up to 256^3. [PK96, PHK+99] use special purpose hardware based on his *Cube-4* ASIC to render a 256^3 volume with ray-casting up to 30 frames per second. In our case, the user is free to resize the workspace's box at any moment, and extend his model wherever he wants, providing *virtually unlimited* grid size. Since the field sampling is regular, two kinds of limitation appear in the current implementation :

- tool too small : the sampling points become too distant relatively to the tool size. The tool isn't correctly sampled, and artifacts due to noise appear.
- tool too large : the tool covers so many sample points that their update is no longer possible at interactive rates.

We report in the following table some statistics concerning differently sized *toothPaste* tools adding some material to the object represented in Figure 2.2. This object corresponds to a `cornersTree`, `cubesTree` and `edgesTree` having respectively 15573, 19791 and 4200 elements. The corresponding balanced binary trees respective depths are 14, 16 and 13. The iso-surface displayed has 4200 vertices and 8392 triangles. The application runs on an SGI *Onyx2/IR* with a 195MHz R10k processor and uses 12 and 10 MBytes to store the structures with the balanced binary trees and hash-tables implementation. For each tool size, we report an average frame rate (wall-clock time) and some results concerning the number of corners and cubes covered in a *best* and *worst* case, depending of the tool's local bounding box orientation.

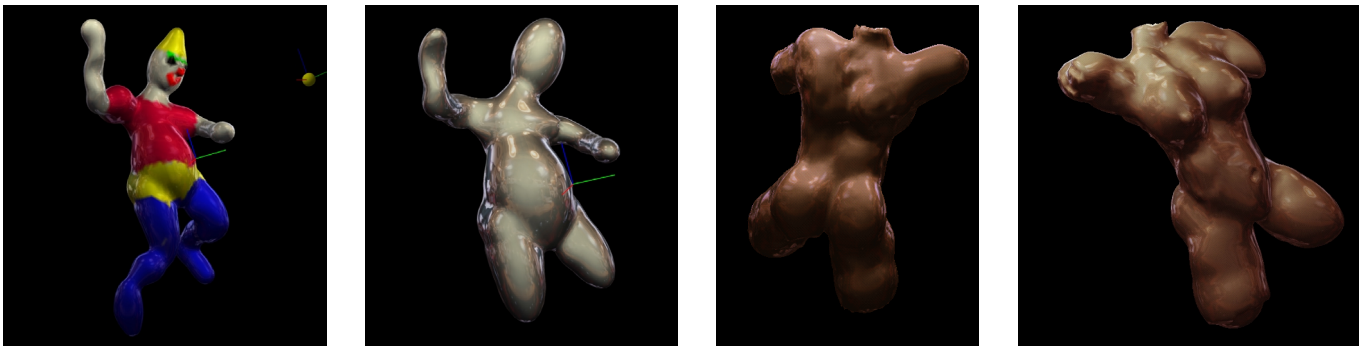
	balanced binary trees implementation				hash-tables implementation			
frames/s	#corners visited	#corners computed	#corners dirtied	#cubes treated	#corners visited	#corners computed	#corners dirtied	#cubes treated
19-23	216 1331	125 203	93 **110**	184 209	1200 4590	891 890	470 **463**	722 707
7-8	1452 4352	887 975	501 **508**	751 771	3825 11520	3136 3012	1587 **1571**	2130 2104
3-4	2744 8316	2197 1919	1021 **1003**	1424 1407	6156 18144	4608 4608	2405 **2417**	3107 3125

This shows that since we are able to rapidly reject the corners lying outside of the local tool's bounding box, the tool orientation isn't affecting the field update performance. This also reveals that the hash-table implementation is more effective : between 2.5 and 4 many more `Corners` updated at equivalent rates.

We show in the following some sample sculptures produced with our application. To give some order of magnitude, the *Bust* model displayed in Figures 2.14.c and 2.14.d corresponds to 54066 values (i.e. `Corners`), 64039 `Cubes` and 9566 `Edges`. The iso-surface is made up of 9566 vertices and 19128 triangles and the programs uses around 25 MBytes to store it.

(a) (b) (c)

FIG. 2.13: Using tools designed inside the application : (a) building the tool. (b) shape modeled with the preceding tool. (c) sample *sculpture* built in a few minutes using a finger-shaped tool.



(a) (b) (c) (d)

FIG. 2.14: (a) A clown character built using only ellipsoidal tools : running the program on an SGI O2 workstation, it took around one hour and a half, including lots of trial and errors. (b) the same clown at an earlier building stage. (c) and (d) show a bust modeled with the same settings in around 4h : its simple display (without any field modification) is achieved on the O2 workstation at 2.7 frames per second (with 2 point light, and without texture nor display list).

7 Future work

There are still some improvements to conduct concerning the visual quality, such as enhancing the stereo display, or adding some visual cues such as shadows (either with textures, or volumes) or depth of field.
Another key feature that will definitely improve the immersion of the application into reality is force feedback : a first idea was proposed by [AS96, Avi98].
An important limitation in our current implementation is the fixed spatial sampling resolution of potential fields. At the moment, we are planning to use octrees to store field values, but a multigrid approach also looks promising.

CHAPITRE 3

Second prototype

Ce chapitre reprend l'article soumis à publication dans Graphical Models en mars 2001, accepté pour publication en avril et à paraître dans le numéro spécial sur Volume Modelling courant 2002.

Abstract

We propose a sculpture metaphor based on a multiresolution volumetric representation. It allows the user to model both precise and coarse features while maintaining interactive updates and display rates.

The modelled surface is an iso-surface of a scalar-field, which is sampled on an adaptive hierarchical grid that dynamically subdivides or undivides itself. Field modifications are transparent to the user : The user feels as if he were directly interacting with the surface via a tool that either adds or removes "material". Meanwhile, the tool modifies the scalar field around the surface, its size and shape automatically guiding the underlying grid subdivision.

In order to give an interactive feedback whatever the tool's size, tools are applied in an adaptive way, the grid being always updated from coarse to fine levels. This maintains interactive rates even for large tool-sizes. It also enables the user to continuously apply a tool, with an immediate coarse-scale feedback of the multiple actions being provided. A dynamic Level-Of-Detail (LOD) mechanism ensures that the iso-surface is displayed at interactive rates regardeless of the zoom value ; surface elements, generated and stored at each level of resolution, are displayed depending on their size on the screen. The system may switch to a coarser surface display during user actions, thus always insuring interactive visual feedback.

Two applications illustrate the use of this system : Firstly, complex shapes with both coarse and fine features can be sculpted from scratch. Secondly, we show that the system can be used to edit models that have been converted from a mesh representation.

2 Introduction

Sculpting with direct 3D interaction has attracted much attention in the past few years. These approaches provide the user with direct interaction with a 3D model via a tool linked to a 3D input device ; this may range from a virtual trackball attached to a 2D-mouse to a force-feedback articulated arm (see Figure 3.1). Mid-range input device include 3D-mice, such as a Spacemouse or a Spaceball. The user perceives the sculpted object on a classical screen with or without stereovision, or with semi-transparent stereo-glasses. The targeted user of these systems should already be familiar with 3D interaction such as clay-modelling or real-sculpting. This is contrary to users of 2D metaphors such as Teddy [IMT99], where 2D curves are used to infer a 3D shape.

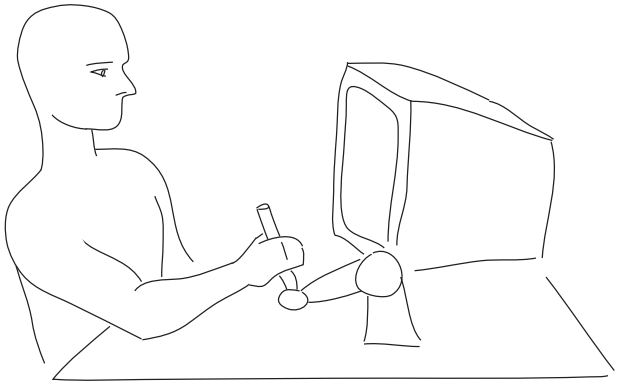


FIG. 3.1: Principle of operation

Hiding the underlying representation is essential in a 3D sculpting system. The user should be able to focus his attention on the shape being modelled rather than on its mathematical, internal representation. This is not the case in traditional, parametric approaches used in CAGD (e.g. the use of a control mesh to edit NURBS or subdivision surface representations). In addition, one of the key features we are seeking is the ability to transparently handle connection and dis-connection of model-parts. To this end, volumetric representations such as implicit surfaces[1] appear very well suited.

Handling topological changes is not the only advantage of implicit surfaces in the context of an interactive sculpting system : they also ensure a *correct* definition of a closed surface which always possesses a well defined interior and exterior. Classical implicit surfaces modelling mostly uses primitives (*skeletons*) that generate a scalar field from which an iso-surface is extracted. These scalar fields can then be combined in various ways, the most basic one being summation. A drawback of this constructive approach is that the cost of field evaluation grows with the number of primitives. If each user action results in a primitive creation, the field evaluation rapidly becomes prohibitive and forbids interactivity. Primitive sorting optimization, such as those proposed by A. Sourin [Sou01] in the case of metal embossing (which requires less primitives than modeling a 3D shape), or other techniques to merge and simplify the primitives could be explored to solve this problem. We rather use a straightforward solution that consists in directly storing a discrete, sampled representation of the scalar field. Various papers have already proposed similar *Discrete Scalar Field* approaches. We discuss them in the next section, before describing our contribution.

2.1 Modelling with discrete scalar fields : Previous work

Interactive modelling based on discrete scalar field representation was early introduced in 1991 by T. Galyean and J. Hughes [GH91]. The field was stored on a regular 3d grid (*voxmap*). The tool used

[1] An implicit surface modelling review is beyond the scope of this paper. The interested reader may find a good introduction in [BBB+97].

to edit the field was also discretized and particular attention was paid to prevent aliasing when the discrete tool was re-sampled into the field grid. Available tool actions included adding or removing *material*, and smoothing the surface through a convolution applied to the 3D field.

In 1995, S. Wang and A. Kaufman [WK95] extended the interaction to carving using tools deduced from a pre-generated 20^3 volume raster or sawing (extruding) via curves drawn onto the screen.

The following year, R. Avila and L. Sobierajski [AS96] used a force feedback articulated arm to command the tool in a similar context. The very rapid update rate required limited the tool size to 3-5 voxels.

We also developed a sculpting systems based on a similar methodology [FCG99, FCG00]. We used a hashing structure to store a regular virtual grid, in order to allow the user to add material without any limitation in space. Other available operations were material removal, and local deformations modelling contact with a rigid tool. We paid very little attention to aliasing, as we considered it as a natural sampling limitation due to the fixed grid resolution. We suggested multi-resolution or adaptive sampling of the field as the adequate solution to the problem. This paper describes this solution.

Surprisingly, a huge amount of contributing papers deal with multiresolution volumetric data or adaptive discretization of implicit surfaces but few multiresolution approaches have been applied in the context of volumetric modelling.

In 1998, J. Bærentzen [Bær98] proposed an *octree-based* volume sculpting system. Used to accelerate ray-casting rendering, the octree is unfortunately static : the subdivision is limited to, and always reaches a fixed *leaf* level, yielding a regular sampling solution very close to the grid used in [GH91]. Dynamic *leaf-node* management to preserve memory in regions of low details or to increase resolution in highly detailed regions was left as a future work.

More recently, A. Raviv and G. Elber [RE00] proposed a different hierarchical approach based on scalar tensor product uniform trivariate B-Spline function. A collection of B-Spline patches with arbitrary position, orientation and size is used to represent the scalar field. The user can create patches and select an active patch to edit with a tool that modifies its scalar coefficients. An additional octree structure is used to sample the collection of patches and conduct a Marching Cubes to extract the iso-surface. The octree resolution is guided by the underlying patches size.

K. McDonnell, H. Qin and R. Wlodarczyk [MQW01] recently presented another interesting approach, based on subdivision solids and surfaces. Three distinct kinds of tools are presented : haptic, geometric and physically-based ; each addressing different features of the model. The paper claims direct physics-based interaction with the sculpted surface to transparently handle the internal subdivision schemes. However, the user still has to explicitly edit control cells to change the topology of the model and to edit mass-points to change its physical behavior. The haptic-based deformation also requires to set a spring connection between the selected mass-point to the tool's cursor.

More recently, Frisken and al. [FPRJ00] proposed a resolution adaptive volumetric approach, named ADF which stands for Adaptively Sampled Distance Field. The basic idea is to use the euclidean distance to a given surface and adaptively sample into a discrete scalar field. Field recomputations due to surface editing are performed either by starting at the bottom of the hierarchy and then performing a simplification pass (bottom-up strategy) or by refining the discrete field where necessary (top-down strategy). Their last paper [PF01a] addressed some limitations of the method such as improving the update rate. In order to maintain interactivity, the update is performed in priority at the neighborhood of the surface. Then distance modifications propagate during idle moments.

This approach presents some similarities with our method, though they were developped totally independently. Both of them exploit the idea of adaptive, multiresolution volume sculpting. However, major differences can be observed in the way the information is stored and sampled in the 3D field. Instead of storing the euclidean distance to the modelled surface, we store a *field function* (similar to the ones used in implicit surface modelling). This field function is the composition of the euclidean distance with a *potential function* that smoothly decreases to zero as the distance grows. This formulation presents several advantages : not only the potential acts as a filter that smoothes

the distance, but, it also bounds the region of influence of editing actions. Consequently only local field updates are required. In contrast, the region where the euclidian distance changes after surface editing is unbounded. So even local editing may produce global field updates in Frisken's approach. Another difference is the subdivision strategy. Frisken stops subdividing when the difference between the distances computed by two consecutive levels of the hierarchy is under a given threshold. With this method, subdivision reaches the bottom of the hierarchy on every distance discontinuity. Such discontinuities occur near each surface concavity, and their spatial extend can be unbounded. Athough subdivision is restricted to cells that cross the surface during edition [PF01a], subdivision in those discontinuity regions may propagate at idle moments. This can lead to over-sampled regions that can be located far away from the surface, severely impacting computational time and memory requirement.

2.2 Overview

This paper presents a 3D sculpture system that enables interaction with a sculpted object, at any modelling scale, without having to be concerned with the underlying mathematical representations. The modelled surface is an iso-surface of a scalar-field. The field is stored in a hierarchical grid with a given subdivision factor. The extent of the structure, in terms of space and resolution, is fully dynamic ; it is driven by the actions of the user and has no size limitation of maximum depth. This system allows for precise, interactive direct modelling through addition and removal of material. As the underlying scalar field representation is completely hidden from the user, the use of the system is intuitive, and gives the feeling of direct interaction with the sculpted surface.

Our main contributions are : (1) an adaptive subdivision of the field guided by the tool ; (2) a progressive update of the hierarchical field using priority queue mechanisms which allow the use of large tools at interactive rates and to continuously apply a tool ; (3) an interactive rendering that directly exploits the hierarchical field representation to locally conduct a Marching Cubes surface extraction and limit the display complexity at each frame according to the camera position.

The remainder of the paper addresses these three points in Sections 3 to 5. Section 6 then shows our results in two different contexts : the direct creation of a complex sculpture and using an editable model that has been converted from a mesh representation. Section 7 presents conclusions and future research directions.

3 Tool Guided Adaptive Subdivision

Our sculpting system lets the user interact with a tool that modifies a scalar field. Our convention is to represent the interior of the sculpted object using positive field values and to have a null field where no *material* exists. This enforces the analogy with the real world, the scalar field being similar to a material density. The displayed surface evolves with a given iso-value of the field, creating the illusion of direct interaction with the object's surface.

Clamping field values so that they lie between a minimum and a maximum value is necessary in this representation. Indeed, as the tools' contributions are applied in a cumulative manner, successive subtractive tool applications would result in a negative field which would artificially prevent new material creation in this region. Symmetrically, successive additive applications would artificially prevent material removal. In our particular implementation, the scalar field is clamped between 0 and 3, the object's surface being defined by the iso-value 1.5.

This section details the constraints we have on the field representation and discusses our choices.

3.1 Data structures

Hierarchical scalar field

The most straightforward, easily edited, scalar field representation consists in sampling the (supposed continuous) field on a *regular grid basis*. Samples of field values are stored, along with other attributes such as the field gradient and color, in a *Vertex* structure (see top-left of Figure 3.3). This can be done without spoiling the field's apparently unlimited spatial extent using a hashing structure to store the grid as in [FCG00]. However, this fixed resolution framework limits the kind of shapes that can be modelled : when a fine grid is used to represent details, surface modification at a coarser scale requires editing too many samples, thus preventing interactivity.

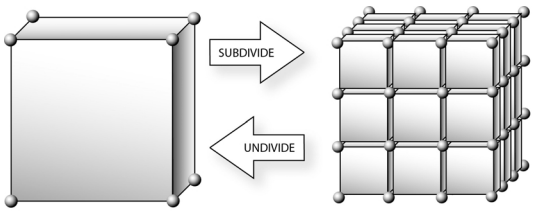


FIG. 3.2: Subdivision principle. Many choices are left, such as replacing the left *Cell* by the subdivided *Cells* on the right or referencing the subdivided *Cells* as children of the left *Cell* (i.e. enriching the left *Cell*), duplicating or not the *Vertex* elements that have the same position, ...

We extend here this approach with the ability to locally refine the sampling rate, such as illustrated in Figure 3.2. In order to preserve the multiresolution representation, the left cell isn't replaced by the sub-cells set, but is rather enriched with it : the sub-cells are its *successors* or *children*. As we do not want to restrict the user to any resolution limit (as well as not restrict the extent of his model in space), we need to allow dynamic creation or deletion of such successors sets. This *a priori* precludes the use of most classical octree storage optimizations. So, we rather try to reduce the structure overcost by allowing a direct jump to much finer resolutions. We subdivide space by a constant factor n on each dimension ($n = 4$ on Figure 3.2). This *ntree* structure resembles the *Recursive Grids* used in bucket-like space partitioning data structures [CP97]. Figure 3.3 (left) gives an outline of the ntree *Cell* structure.

The next point to discuss is whether to : (1) express the samples at a finer level, $k+1$ as a *delta* contribution over the average or median value that would be stored at the coarser level k ; (2) store directly the field value in each sample, thus using simple subsampling for coarser levels.

Solution 1 appears more elegant, as it looks like a wavelet decomposition of the 3d scalar field. However, it yields the extra cost of maintaining the hierarchy coherency. Coarser levels would need to be updated when detailed modifications are conducted on smaller levels, in order to recompute the average or median field's values.

Solution 1 also suggests that large changes on the low-resolution levels could effortlessly be reflected on the higher ones : storing some min-max information along the hierarchy would allow rapid pruning of the volume parts that become completely outside or inside of the modelled shape. However, the hierarchy exploration from the root node to the leaves (which is also requested in solution 2) cannot be avoided, since the surface representation has to be updated.

With the subsampling approach of solution 2, there is no need to compute the interpolated values from the coarser levels : the field value at a vertex is directly given. Moreover it allows **no duplication of** the *Vertex* nodes between resolutions. In contrast, solution 1 would force the eight corners of the *Cell* at the left of Figure 3.2 to be distinct from their counterparts in the sub-cells on the right because

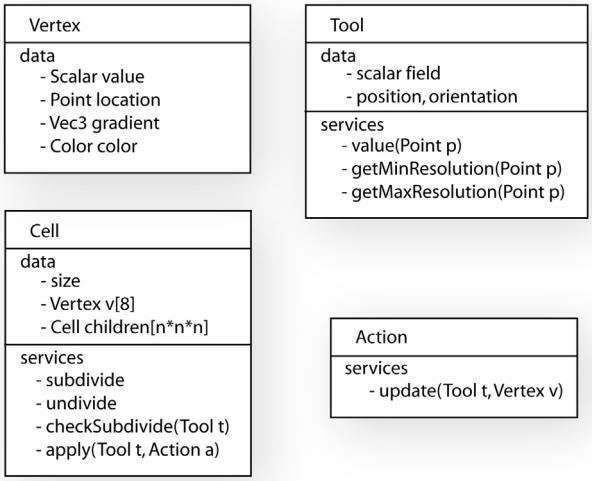


FIG. 3.3: Data-structure used for the field representation (left) and for applying a tool that modifies the field (right).

the sub-cell values define a *delta* contribution over them. Lastly, solution 2 offers a kind of *vertical independence* over the hierarchy : each level is completely independent from its ancestors, and can thus be updated independently. Therefore, we have adopted solution 2.

When subdividing the grid (i.e. during *Cell* creation), we pay special care to share the *Vertex* nodes among common faces or edges between the adjacent *Cells* of the same level. Once these shared structures are wired, their forthcoming updates won't cost more than a time-stamp comparison to prevent useless computations.

What is a tool ?

Basically, a tool is another scalar field that can be positioned, oriented and scaled as the user wants (see Figure 3.3, right). It could be another hierarchical sampled field, or any bounded primitive that can return a scalar value from a given location.

The simplest example (the only one we implemented at the moment in the multi-resolution version of our sculpting system[2]), is an ellipsoidal tool. In our current implementation "material" is represented by positive field values, the field being supposed to be null elsewhere. Our ellipsoidal tool is based on Wyvill's field function :

$$f(p) = \begin{cases} \text{if } d \geq 1 & 0 \\ \text{else} & 3 * (1 - \frac{22}{9}d^2 + \frac{17}{9}d^4 - \frac{4}{9}d^8) \end{cases} \quad (3.1)$$

where p is the query point, and d^2 is the squared distance from the tool center to the point p, **expressed in the tool local frame coordinate**. Translation, rotation, and scaling of this local frame gives the tool's current position and shape. Using Wyvill's field function yields several advantages :

- it is computationally cheap (it uses the squared distance, instead of its squared root) ;
- it has a spatially limited domain of *influence* (i.e. non-zero values) ;
- it is bell shaped, with C^1 variations. This allows smooth blends when different tool contributions are summed.

[2]The use of various tools ranging from simple primitives to user-designed tools was explored in our previous work [FCG00]

The tool is applied iteratively by cumulating its contributions into the sampled field. The modelled surface is the iso-surface at $iso = 1.5$ extracted from this field. As we use the limit of influence volume to display the tool's shape (i.e. the ellipsoïd outside of which the tool's influence is null), the material deposited by the tool always lies inside the tool's volume (see Figure 3.5 left). Continuously applying the tool at the same location progressively extends the created surface, which may eventually reach the tool's border.

Applying a tool

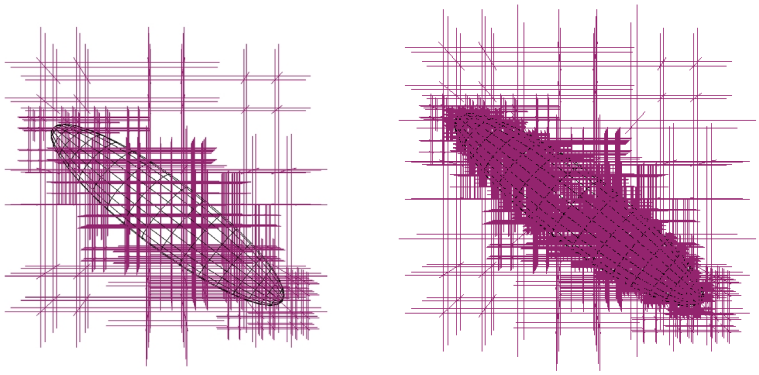
Applying the tool into the scalar field involves two important steps : (1). we must ensure that the tool is correctly sampled, i.e. the cell resolution of the field representation must be small enough to capture the tool's features and (2). for each covered vertex, we have to combine its current field value with the tool's contribution at that point.

The latter is issued using what we call an *Action* (see Figure 3.3, right). An action takes a *Tool* and a *Vertex* as arguments. It computes the tool's contribution at the vertex, and combines it with the current vertex attributes. For example, an *AddAction* simply adds the tool's field value to the vertex field value, uses these values as weights to sum-up the attributes such as the gradient and color, and clamps the field value between 0 and 3.
The algorithm for applying a tool to a cell of the hierarchical field representation is quite simple :

ALGORITHM : Applying a Tool to a Cell

```
Cell : :apply(Tool t, Action a) {
    foreach Vertex v do { a.update(t,v) ; }
    if (I have no children) { checkSubdivide(t) ; }
    if (I have children) {
        foreach Cell child do { child.apply(t,a) ; }
    }
}
```

Now, how do we know if we need to subdivide a given cell (*checkSubdivide* test in the algorithm above) ?
Let's suppose that the tool has an ellipsoidal shape. We would like to obtain something like Figure 3.4, where the sampling rate increases (i.e. the cell size decreases) in regions where the tool has sharp features.



(a) (b)

FIG. 3.4: Sampling an ellipsoidal tool. (a). Only 2 consecutive levels. (b). All the levels hierarchically created.

First, we might query the tool attributes for requirements on a minimal *security* cell-size to reach, in order not to miss any of its features. This could be a global information constant over the tools influence region, or something locally computed. For example, if the tool field is stored as a hierarchy of cells, we could easily use the size of the leaf cell as the minimal size to reach.
Once we have reached this minimal size, the field could still be ill-sampled. For example, if we use a subtractive action, we might create discontinuities, even with spherical tools. Our choice here is to try to estimate the *discrepancy* of the field. If the cell's discrepancy is higher than a given tolerance, we go on subdividing (see Figure 3.5). Pragmatically, we use this strategy only on cells that are crossing the surface, as this is our region of interest (see section 5.1 for more details on the discrepancy estimator). Moreover, using this strategy in regions where no surface exists could disturb the surface when it reaches these regions ; as the details existing only in the field (and consequently hidden from the user) would suddenly become visible on the surface being created.

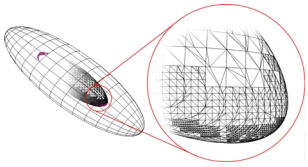


FIG. 3.5: Sampling an ellipsoidal tool : the large ellipsoid is the tool, and the small one inside it is the surface created. The figure shows the maximum resolution reached in highly curved areas.

As stated above, using a subtractive tool can cause discontinuities in the scalar field so the subdivision process might never end. Here again we rely on the tool to query a *maximum depth* to reach. In fact, this is formulated as a smallest cell size not to overpass, which we call the *maximum resolution*. It could, exactly as the *minimum resolution* above, be locally adapted inside the tool's region of influence. At the moment, our ellipsoidal tools only expresses it as a constant factor, depending on the tool's scale used.

This leads to the following algorithm :

ALGORITHM : Testing a cell for subdivision

```
Cell : :checkSubdivide(Tool t) {
    if (size > t.getMinResolution() ) {
        subdivide() ;
    } else {
        if (size < t.getMaxResolution() ) {
            if (estimateDiscrepancy() > acceptableDispcrepancy) {
                subdivide() ;
            }
        }
    }
}
```

We do not have *a priori* knowledge of the field's profile before we reach the bottom level of subdivision. Thus we can not stop the subdivision to conduct any simplification ("undivide" of Figure 3.2) at any particular level, as we can not guarantee that the finer level will not add surface details. As a result, we conduct a separate simplification pass over the whole sampled field at idle moments of the interaction. The simplification strategy we are currently using is rather drastic, as it destroys every *Cell* that does not (and whose children do not) cross the surface.

Updating the coarser levels first, as depicted in the **apply** algorithm, is crucial to provide an interactive visual feedback. This means careful initialization of the newly created *Vertex* set. Figure 3.6 represents a slice of the scalar field value for a cell that is to be subdivided. Figure 3.6.b shows the example effect of applying a tool with an *Add Action*. The value of the two border vertices of the cell is modified with the tool's contribution. The creation of the new vertices for the sub-cells requires interpolation of the field value **prior** to the current tool application, as illustrated in Figure 3.6.c. so that the tool's modifications can properly be added.

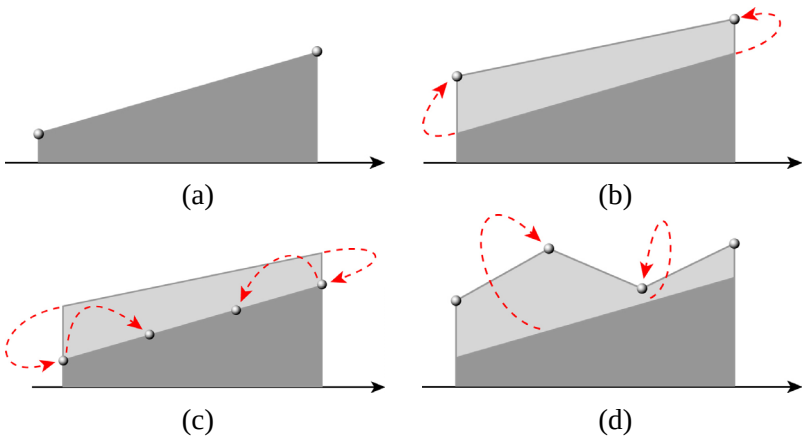


FIG. 3.6: Applying a tool at level k (1D representation). The figures represent the field value as a function of vertex location : (a). The field before any modification. (b). Updating vertices at level k. (c). Creating the level $k+1$ and initializing it with the values interpolated from level k **before** the update. (d). Updating vertices at level $k+1$

We could simply use a kind of *reverse action* to obtain these values and attributes back from the current ones. Unfortunately, as the field values are clamped, the tool application is not reversible. So we have to explicitly cache the field values prior to current tool/action modification.
Our current implementation of this *Vertex cache* uses *STL* hash_map indexed by the memory addresses of the *Vertices* to map the Vertex state prior to the current tool's modification. We thus have one *Vertex cache* per couple Tool-Action application.

As a side effect, this provides almost enough information for undoing the effect of the currently applied tool.

Undo

Undo files[3] are directly generated from the *vertex cache* built during the field update. These files are simply a dump of the set of *modified* vertices. The memory addresses used to retrieve the vertices are not stored as the vertex might be destroyed or reallocated between the save and the reload of this undo-file.

We decided to treat *Undo* as a special *Action*. The first consequence of this is that we can easily use the current tool's contribution to *weight* our undo-action. We call this process a *Progressive Undo Action*. On the same basis, we can simply use the tool's contribution as an *in/out* selector to activate or not the undo, we call this a *Local Undo Action*. We also provide the commonly expected *Undo Action* behavior, i.e. a global undo, which is independent from the current tool being used.

[3]The use of undo files does not forbid interactivity. We suspect that this may be thanks to the Operating Systems filesystem caching. Both under Irix, Linux and WindowsNT/2000, this appears to work surprisingly well.

The second consequence of treating *Undo* as an *Action* is that, as the undo-action is handled internally as a normal tool action, it transparently generates the *redo*.

4 Priority Queue Based Field Update

The recursive approach for performing field updates, described in Section 3.1, isn't suitable for an interactive update. Actually, it walks along the hierarchy depth first, thus missing the requirement to update coarser levels before considering finer ones. Next, we explain how to get this feature, which is essential for achieving interacive display.

4.1 From coarser to finer levels

We use a priority queue sorted on the level addressed and on the tool/action concerned to ensure an update from coarser to finer levels. The tools/actions must be sequentially applied, but we should update the coarser levels first. Thus we perform a straightforward priority evaluation based on these two criteria.

Our implementation uses a *STL* priority queue, which itself relies on a *STL* vector by default. The level is simply the cell size and the tool/action couple to apply is enclosed in a local *ToolCopy*. The *ToolCopy* is a *frozen* copy of the tool at its application time. It contains some state information (such as position, orientation, scale, color, etc), a reference to the tool and action involved and a unique identifier (an integer, acting like a time counter, see Figure 3.7) that also serves as a time-stamp to avoid useless computations.

The queueing internals are enclosed in a *Level Manager* (Figure 3.7). It initiates the application process from a *Tool/Action* couple by creating a *ToolCopy* from them and inserting it with the current root-cell's size into its priority queue.

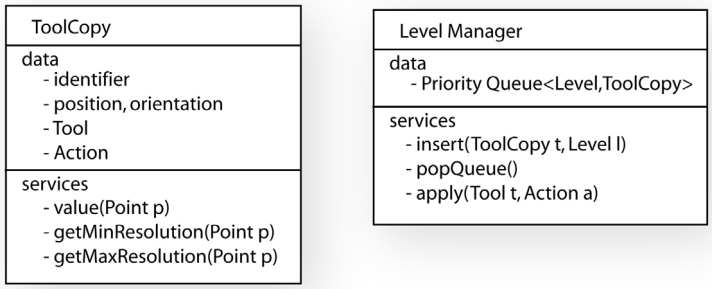


FIG. 3.7: Data-structure used for priority queues handling.

The *Apply* procedure outlined in 3.1 is not altered much. It still updates its internal vertices and subdivides if needed. Then, instead of recursively calling apply on the existing children, it simply inserts a new element made up of the same *ToolCopy* and the next level.

4.2 Emptying the queue

To empty the priority queue we need to find all the cells of a given size (or level) that are *intersected* by the *ToolCopy* (which is much like an image of the *Tool* at the moment its application was posted). Without any additional structure, this would mean recursively walking through the cells hierarchy from the root-cell until we reach the cells having the desired size. To the cost of walking

from the root-cell we must add the extra-cost of the intersection test with the tool for each cells of the intermediate levels.

To avoid these useless computations, we use a simple *cell-queue* with basic constant-time operations (pushback and popfront) to temporarily store the cells intersected by the tool from one level to the next. Cell-queues are indexed by a *ToolCopy* and the size of the cells it contains. They can be directly inserted/handled inside the manager priority queue, whose elements are then the cell-queues (Figure 3.8).
The *Apply* procedure of 3.1 is again slightly modified : it receives a cell-queue as an extra parameter. Children cells that intersect the tool are appended to this queue.

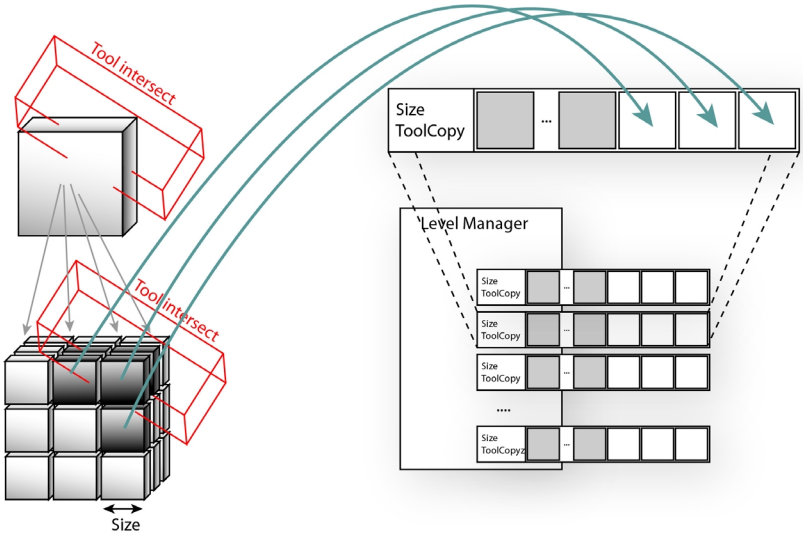


FIG. 3.8: The *Level Manager* : a priority queue of cell-queues.

Another benefit of these cell-queues is that they allow **interruption of** the processing of a given level if any coarser level is inserted inside the manager. The interrupted cell-queue is simply re-inserted in the manager priority queue, and is properly handled from where it was suspended when the working task returns to it.

4.3 Multi-process

A key feature to ease interactivity is to have an interaction/display thread running *separately* from the background update process. We considered and tried various multi-threaded strategies but finally a two-threads approach proved to be the most simple and efficient. It is simple because we only have one process modifying the data structure so little exclusion mechanism is needed. It is efficient for the same reason : no system call overhead or blocking/deadlock situation.

The *update-thread* is created at program start-up and empties the priority queue of the *Level Manager* in the background. The user interacts in a separate thread. Applying a tool becomes the insertion of one Cell-queue that contains only the current root-cell into the *Level Manager*. The *update-thread* then possibly interrupts its current cell-queue processing and handles normally the next sub-cells-queues insertion and processing.

The only blocking access concerns the insertion and removal of cell queues in the *Level Manager* priority queue. It is easily handled via a classical locking mechanism. It is important to note at that point that the update process gets most of its workload to empty the specific *Cell queue* it extracted

from the *Level Manager* and filling one for the next level. Consequently, the access to the *Level Manager* to insert or retrieve a *Cell queue* is not really blocking as both processes do not intensely access it. Note also that the update process is the only one accessing the Cell-queues it empties or fills, as the draw process accesses the Cells through the hierarchy and not through the Cell-queues. So inside each Cell-queue no exclusion mechanism is needed.

We implemented the *threading/locking* facilities both with Posix threads, IRIX sprocs, and Windows threads with no remarkable speed impact. When the priority queue of Cell-queues to update becomes empty, the update thread conducts the simplification step evoked in section 3.1. Once this simplification is achieved, we use the blocking facility of the lock system calls to suspend the update thread and save processor resources.

4.4 Flexibility

This priority queue mechanism appears very flexible. Though we are not exploiting this at the moment, it could handle several tools acting on the field at the same time without requiring any modification. These tools, possibly controlled by different users, would be inserted into the *Level Manager* as usual and transparently handled by the update-thread. The tools could even interact in overlapping regions as they would be properly inserted into the priority queue thanks to the exclusive insertion mechanism.

More pragmatically, priority tuning could also help improving the interaction quality. Suppose the user makes a large change at some large low-detail levels of the cell hierarchy, then comes to another region where small details already exists, and tries to edit these small details. With our current priority scheme, no update will happen until the global update sequence level reaches this level. We could easily avoid this problem with little impact on the rest of the modules, by tuning the priority evaluation according to the user's current focus.

5 Rendering and Simplification

Excepting [WK95, AS96, Bær98, PF01a] that use ray-casting to visualize the iso-surfaces of the scalar field, all other approaches are based on the Marching Cubes Algorithm.
Basically, the ray-casting process allows an update of only the portion of the screen that is being edited (tool projection footprint). This may appear as more efficient. However, it also forbids movement of the object while editing it because the redraw effort would then become too important.

Our first approach [FCG00] did confirm that the Marching Cubes algorithm was well suited to interactive update and visualization of the surface, exploiting graphics hardware. In our resolution-adaptive sculpting system, we still use the costless sphere-mapped environment texturing that was introduced in [FCG00] to improve shape perception by generating high quality highlights.

5.1 Surface creation

To display the iso-surface, we test each cell against *iso-crossing*. This consists of comparing the eight corners' field value to the iso-value. These comparisons serve to compute a Marching Cubes configuration index. If the cell *crosses* the iso-value, we associate a *Surface Element* to it. This structure stores the Marching Cubes configuration index (an integer) and at most twelve pointers to some *Surface Points*. The *Surface Points* are the intersections of the iso-surface with the current Cell's edges. They are linearly interpolated from two adjacent *Vertices* to match the iso-value. They also interpolate the vertices attributes, such as the gradient (that becomes the surface normal) and color

information. Simple *time-stamping* mechanisms avoid multiple computations of these points during the vertices update step through the Cells exploration.

Additionally, the *Surface Element* is used to estimate the surface *discrepancy* introduced in Section 3.1. We need a quantity that indicates the *flatness* of the extracted surface. We decide to exploit the normals extracted at the surface points. If the normals are all pointing in a similar direction, the surface will be well represented by our linear approximation. On the contrary, if they have very different directions, our linear approximation is poor and the sampling rate should be increased to better match the underlying iso-surface. We use a straightforward estimator that computes a kind of standard deviation of the surface normals (see Figure 3.9).

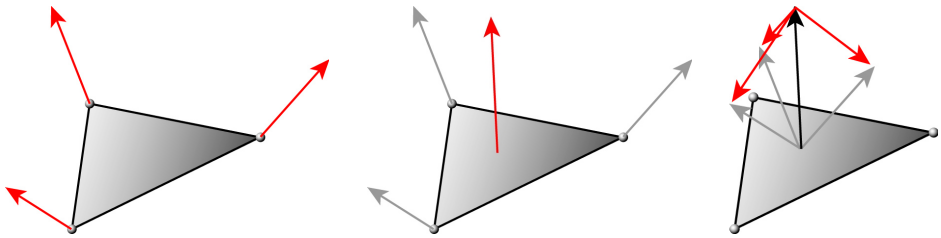


FIG. 3.9: Estimating the *flatness* of a surface element : we first compute an average normal of the normals computed at the surfaces points (b) and then sum-up the squared length of the difference vectors between the surface point normals and the average (c).

As a result, we obtain many approximations (*Level Of Detail*) of the iso-surface at each level of the cells hierarchy. Figure 3.10 shows, for the same surface, different approximations that are computed and stored along the cell hierarchy.

5.2 Surface display

The refinement process guided by the discrepancy estimator enables correct sampling of the field. However, at the leaf level of the hierarchy we obtain far too many triangles for any current graphic hardware to display interactively.

Our first solution to remedy this was to conduct a simple object based view frustum culling. As the cells hierarchy also constitutes a good space partition, we can efficiently prune the cells outside of the view frustum. This is particularly useful when editing small features that are part of a large model. For example, most of the surface of the sculpture displayed in Figure 3.12.(a) gets culled very early in the hierarchy, thus the hierarchy exploration for displaying the surface is concentrated on the visible parts.

However, this is clearly not sufficient when the whole model is contained inside the view frustum. To address this problem, we compute for each cell an estimated projected size on the screen. It is estimated from the cell's size and the distance of the cell's center to the screen projection plane. Using this *projected size*, we can stop the hierarchy exploration when the projection of the current cell becomes too small. For example, if the projected size of a cell is smaller than a pixel, the triangles contained inside its children will be smaller, so we avoid visiting them and rather draw the surface element of the current cell.

This mechanism gives control over the number of displayed cells at each frame and dynamically selects a LOD dependent on the distance to the projection plane. We first exploited this display complexity control with a global *minimum projected size* that the user could edit (see Figure 3.11). Nevertheless, this fixed control was not sufficient because even during idle moments, the displayed

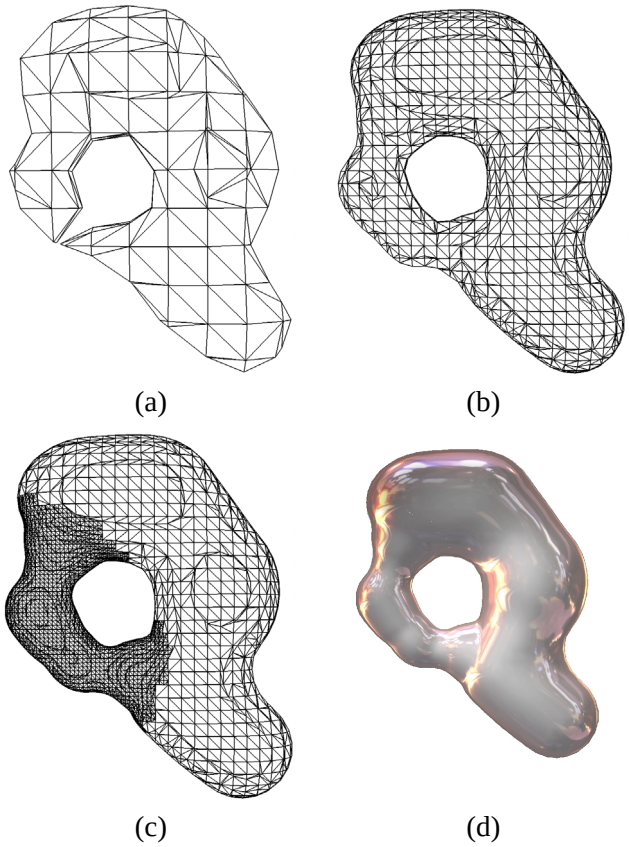


FIG. 3.10: Different levels of surface elements from coarse (a) to fine (c), with an intermediate level (b). Figure (c) illustrates the adaptive refinement of the sampling rate : the wireframe surface displayed reached the bottom of the hierarchy. (d) shows a textured version of the polygonal approximation in (c).

surface remains unrefined. Moreover, if the user zooms in or out, the surface can result in a model that may be poorly or over populated.

Our solution here is to automatically adjust this *minimum projected size* from frame to frame to maintain a given framerate during user interaction. At the end of each frame, we measure the time spent from the previous frame end and we use the difference with the desired *display time* to weight the *growing factor* of the *minimum projected size*. Pragmatically, we used the third power of this difference to minimize its influence when the display time is near its goal, and emphasize it when it's far from it, keeping its sign. When the user is idle, the limit projected size is progressively reduced close to zero. So the fully detailed geometry can be rendered if the user waits sufficiently long ; which will allow a non-interactive but accurate display during the session.

Contrary to other multiresolution iso-surface constructions, we pay no attention to the cracks that appear between adjacent cells of different sizes. Solving this problem by drawing more triangles would not be a solution, since the problem may come from the fact that the field is not defined at the same resolution in adjacent cells. A solution would be to simply reconnect the surface-elements found at each cell. We deliberately choose to NOT to do this. A first reason for this choice is that the surface is always changing during the sculpting process. Another reason comes from the fact that we dynamically select, at each frame, where to stop in the cells hierarchy display. Reconnecting surface elements would force us to always track the neighboring cells. This would largely slow down the display rate, which is especially true in our unconstrained hierarchy (adjacent cells could be distant from more than one level of resolution). Moreover, as long as a sufficiently large number of polygons is displayed, the cracks can remain hardly visible (see Figure 3.10 (d) for example), it is thus *a*

posteriori not worth the effort.

If needed, performing a global offline reconnection process would be easy if the current surface has to be converted to a standard mesh representation to output an export file.

6 Applications

We show here two examples of sculptures produced with our system. The objects were sculpted using a 2D-mouse with a ”virtual trackball” metaphor (a Magellan spacemouse is also available). The images are direct screen shots from the interactive sculpting sessions.

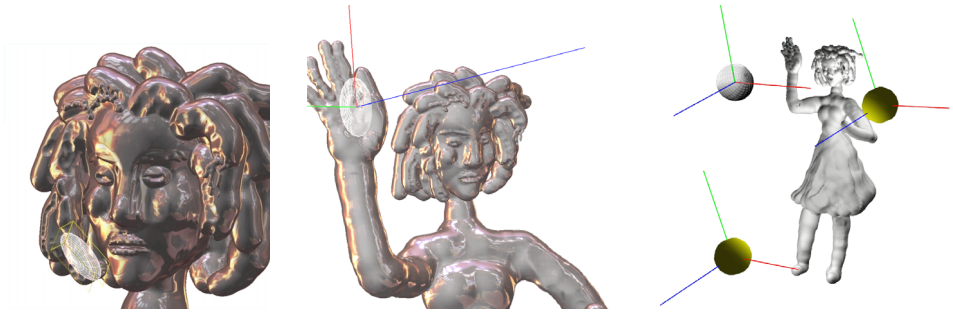
Additional material, such as higher resolution images and a video showing our system interactivity are available at
`www-imagis.imag.fr/Publications/2001/FCG01`.

6.1 Creation from scratch

Our first example is a character model created from scratch. The wireframe views in Figure 3.11 illustrates the display complexity control. In this case, we use a fixed *minimum projected size*, so the closer the model, the more detailed surface version we get. If the *minimum projected size* was automatically updated, as no part of the surface is culled, we should obtain approximately the same complexity on the four views.

FIG. 3.11: The different views show the adaptation of the LOD while the model is moved closer to the camera.

Figure 3.12 shows some steps of the modelling process. Face details have been modelled first using additive tools of different sizes to create the head, the chin and the hair (see Figure 3.12 (a)). Then, a negative tool has been used to remove material in the eye regions and the eyes have been created inside these hollows by successively adding and then removing material. High resolution editing was necessary for sculpting the lips. Next, a coarse body was progressively created (Figures 3.12 (b) and (c)). (b) shows details on the hand creation, which uses again successive addition and removal operations. The sculpture was created by a beginner with this multiresolution sculpting system in about two hours.



(a) (b) (c)

FIG. 3.12: Three steps of a character's modelling. Figures (a) and (b) show two closed views used for modelling respectively the face and the hand. Figure (c) shows the tool and two lights (represented as spheres) around the character, and uses a different rendering style, with no high lights. Note the different surface resolutions in the three views, which are quite apparent in the eyes region.

6.2 Editing of a model imported from a mesh

The ability to import and edit existing models is an interesting feature.

Volumetric dataset are easily converted to our *material density* 3d field representation : it consists of performing a translation and scale of the input values to make the iso-surface match the desired iso-value, and the field values range in the desired min-max interval.

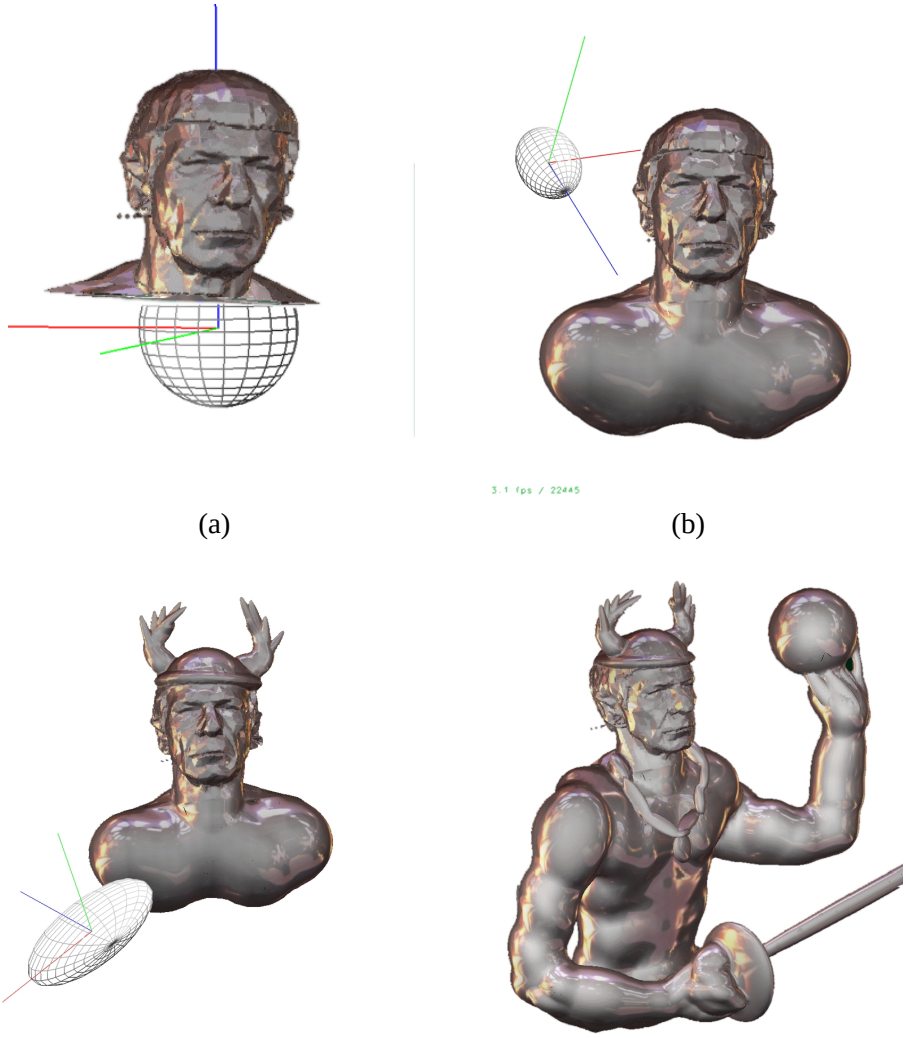
Importing polygonal models can be cumbersome, especially if the input models are non-manifold/orientable or contain some intersecting polygons. We initialize the field on each grid point using the signed distance to the polygonal mesh. Field values are then translated and scaled to fit the desired iso-value and interval. A detailed description of this conversion method is beyond the scope of this paper, but note that the cell hierarchy can be of great help during the conversion process, since going deeper in the hierarchy can help disambiguate intricate situations.

Figure 3.13 shows an example where we edit a converted polygonal model. Large scale additive tools have been used for creating the main character's body features while preserving surface smoothness. A spherical tool inserted inside the head was used to create the helmet, and a slightly larger flat version of it for the helmet's border. The wings were created by successively using very small additive tools, thus creating high resolution details. The same technique was used for the chain. Creating the whole character took one and a half hour.

7 Conclusion and Future Works

We have presented a 3D sculpture system that provides direct interaction with the model – an iso-surface of a scalar field – at any modelling scale. The field representation, totally transparent to the user, is a fully dynamic hierarchical structure, that refines where and when needed. Interactive rates are obtained whatever the modeling scale due to a progressive update of the hierarchical field, from coarse to fine resolutions, using a priority queue mechanism. This mechanism enables continuous application of a tool while maintaining interactive update rates. Rendering relies on the hierarchical structure for performing local iso-surface extraction and displaying the adequate LOD depending on the size the region occupied on the screen. We have shown that our system can be used for both direct creation of a complex surface, or for editing a surface converted from another representation. Our examples also show that our system is especially well adapted to the design of organic shapes, such as the creation of virtual characters.

A first extension to the system would be to interface it with a force feedback device, such as existing articulated arms (i.e. *Phantom*). One should note that combining such an haptic system with our



(a) (b)

(c) (d)

FIG. 3.13: Editing a model imported from polygonal data.

multiprocessing technique based on pthreads is not a problem, since the phantom interface is running in a specific thread under IRIX OS. We have already experimented with it in the previous version of our sculpting system [FCG00] as described in [Bla00]. Haptic interaction proved to be a great aid in the sculpting process, since the user can “feel” the model, and thus decide more easily if he is adding material onto or in front of the surface. Extending force feedback to the multi-resolution version of our sculpting system would however require changing the way the feedback force is evaluated. Similar to the rendering pass, the cell hierarchy could be used to decide at which level force evaluation is performed according to the available time-interval (usually, force feedback is computed at 1000 Hz). This would result in coarser forces when large tools are used, but real-time response would be preserved.

However, haptic interaction through a force feedback device is still very far from the sense of touch a designer feels when he sculpts with real clay. We are currently planning to use a more direct “real-object interface”. The user would manipulate a real deformable object, serving as an avatar for the totality, or for a part of the sculpture. His hand gestures would be captured by cameras and the reconstructed gestures would be used to deform the virtual sculpture. Our hierarchical representation may be very useful in such a framework, since the ability to maintain interactivity while combining several tools and actions is essential for simulating interactions with the ten fingers.

Other interesting extension would be to extend from multi-tools to multi-users, maybe over a network. The manager mechanism seems flexible enough to be extended to multi-managers without enormous effort. One straightforward solution would be to use a *master manager* to handle and order the tools requests for application, and then dispatch them to the *slave managers*. These *slaves* would locally maintain a discrete field's copy, thus limiting the network traffic to a simple *ToolCopy*. This would enable increase efficiency in a collaborative sculpting task by increasing the number of artists ; the lack of collaborative design facilities often being cited as one of the main limitation of digital models [Wes98].

CHAPITRE 4

Développements complémentaires

Dans cette partie, nous allons aborder les travaux non publiés dont certains ne sont que partiellement implémentés. C'est donc à la fois la présentation de développements complémentaires et une réflexion sur les deux modèles proposés. Les parties sur le retour haptique et les opérations de sélection, copier/coller proviennent du travail effectué avec Renaud Blanch lors de son stage de deuxième année Supélec durant l'été 2000. Ils ont été implémentés uniquement dans le premier prototype monorésolution présenté, mais les idées introduites sont relativement indépendantes de la représentation choisie pour stocker le champ scalaire.

1 Retour haptique

Dans la cadre de cette thèse, iMAGIS a fait l'acquisition d'un bras articulé PHANTOM Desktop de Sensable (Figure 1.90.(a)). Ce dispositif correspond parfaitement à notre métaphore de sculpture virtuelle, dans le sens où l'utilisateur agit directement sur la surface qu'il construit via un outil.
Tout comme nous n'avons pas cherché à effectuer une simulation par modèle physique du comportement du matériau pour la modélisation, notre objectif ici n'est pas de simuler une force correspondant à un matériau réel donné. Nous avons plutôt adopté une approche qualitative, visant à exploiter ce dispositif pour aider à appréhender la forme et faciliter son édition.

Sensable commercialise par ailleurs un logiciel de sculpture annoncé durant le salon Siggraph en été 1998. Le logiciel de sculpture *FreeForm* propose une interaction où l'outil est à l'intérieur de l'objet, et on le tire pour faire une excroissance. A l'inverse, lorsque l'outil est à l'extérieur, on pousse pour effacer de la matière. Nous avons préféré une approche où l'outil demeure toujours extérieur à la surface sculptée, et le choix d'ajout ou retrait de matière se fait plutôt par un changement de mode ou d'outil.

Le bras à retour d'effort est livré avec un kit de développement, appelé GHOST, permettant de l'exploiter à la fois sous Windows et Irix. Cette bibliothèque offre tout un graphe de *scène haptique* traitant les modèles polygonaux et offrant des éléments d'interfaces comme des boutons, des valua-

teurs ou des manipulateurs possédant un comportement, visant à standardiser l'interface et à faciliter le développement. Il est possible d'associer des propriétés *physiques* aux surfaces polygonales comme une masse, un coefficient de friction et une dureté. Ces derniers correspondent à priori à un modèle masse-ressort avec une composante tangentielle pour la friction [MS94].

1.1 Exploitation du modèle polygonal

Comme on dispose pour l'affichage d'une représentation polygonale de la surface sculptée, la manière la plus simple d'implémenter un rendu haptique de la forme sculptée pourrait sembler de profiter du kit de développement GHOST pour la gestion bas niveau des propriétés de surface.
Cette approche nous fait par ailleurs bénéficier des développements spécifiques pour pallier les inconvénients des représentations polygonales, comme l'utilisation de points de contact (ou *proxy*) et l'interpolation de normales [ABM+99] (Figure 4.1).

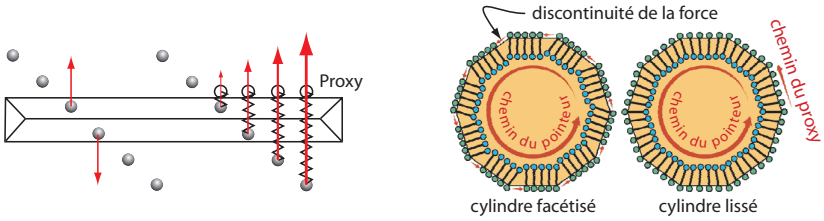


FIG. 4.1: (a) Illustration du concept de *proxy* pour contourner les problèmes de transpercement ; (b) montre le principe d'interpolation des normales pour réduire les discontinuités perceptibles dans la force (images à partir de [Avi98]).

Malheureusement, notre surface n'est pas statique : elle évolue constamment, ce qui interdit le traitement global de toute la surface par le kit de développement. En effet, le fonctionnement du bras requiert une fréquence de mise à jour de la force de l'ordre du kilo Hertz. Les tests de collision avec un modèle polygonal doivent donc être optimisés grâce à une partition spatiale précalculée. Il n'est pas prévu dans GHOST de mise à jour de la surface, à fortiori de la partition.
Nous ne sommes pas les seuls et les premiers à vouloir traiter des surfaces déformables avec GHOST. Par exemple O. Körner et al. dans [KSW+99] cherchent à explorer des données médicales volumiques. Dans ce but, ils extraient localement une surface à partir des données volumiques et utilisent ce modèle local comme représentation pour GHOST.

Notre stockage du champ potentiel scalaire constitue déjà une bonne partition du modèle. Il est donc facile, connaissant la position de l'outil d'extraire de son voisinage les cubes contenant un élément de surface. Comme le retour d'effort calculé à partir des triangles est ponctuel, il n'est pas nécessaire de collecter un grand nombre de triangles ou de tenter de les faire correspondre à l'intersection de l'outil avec la surface. Dans notre cas, la consultation des 3^3 cubes entourant le pointeur a suffit. La Figure 4.2 montre les triangles ainsi collectés et qui sont envoyés à GHOST. La mise à jour de cette représentation locale se fait à 300Hz sur une Onyx2 équipée de processeurs R10000.

Nous n'avons toutefois pas poursuivi dans cette voie, car pour une raison inexpliquée, la restitution obtenue donnait la sensation de surface *collante*. Lorsque le stylet était entré en contact avec des triangles, il fallait exercer une force non négligeable pour s'en éloigner. Nous n'avons pu trouver l'origine de ce problème, sans doute lié à une utilisation de GHOST inhabituelle. Nous avons été confortés dans la recherche de solutions alternatives par les problèmes évoqués dans [KSW+99]. Ils

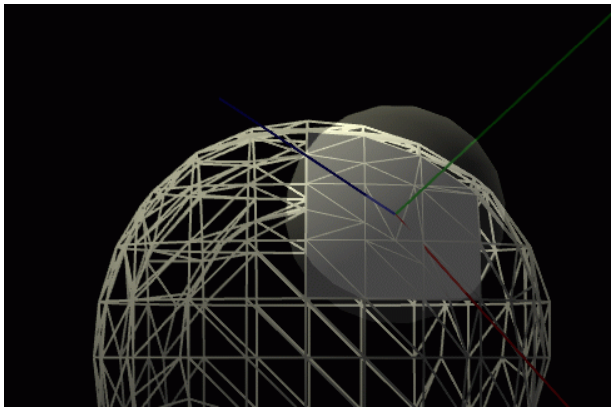


FIG. 4.2: Exemple de triangles sélectionnés pour le rendu haptique. L'objet est en fil de fer et l'outil en transparence.

reportent en effet des problèmes de gestion mémoire *sans doute* internes à GHOST, empêchant leur application de fonctionner plus de quelques minutes (le temps d'épuiser les ressources systèmes). La version courante de ce kit est 4.0 pour Windows et 3.1 pour Irix. A l'époque de nos implémentations (été 2000), la version la plus à jour était la 2.1. Il se peut que ces problèmes soient arrangés dans les versions plus récentes de GHOST.

1.2 Champ potentiel scalaire

GHOST fournit également un accès bas niveau permettant de lire la position du stylet (translation et orientation) et de spécifier une force à renvoyer en son extrémité. Pour éviter toute interruption / discontinuité dans la force restituée par le bras, GHOST utilise une boucle synchrone à 1KHz, en utilisant les extensions temps réel d'Irix pour maintenir fortement cette contrainte. Comme l'exploration de notre structure, aussi bien celle reposant sur les tables de hashage que celle utilisant un octree, ne permet pas d'évaluer le potentiel pour la position du pointeur en un temps constant, nous avons utilisé une autre boucle, asynchrone cette fois-ci, dédiée à la localisation du pointeur et au calcul de la force uniquement.
Nous avons donc trois processus s'exécutant en parallèle : un pour l'affichage et la mise à jour du champ potentiel, un pour le calcul de la force, et un pour l'envoi de cette force et la récupération des positions avec le bras articulé PHANTOM (voir Figure 4.3). La répartition éventuelle sur plusieurs processeurs est laissée au système d'exploitation.

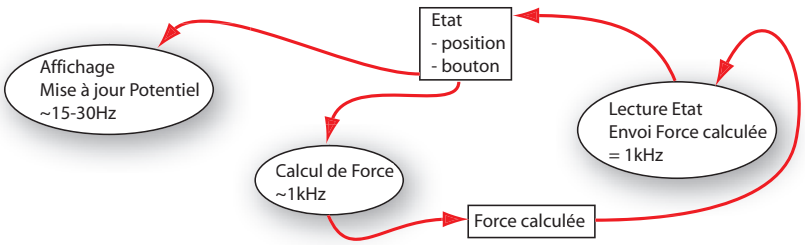


FIG. 4.3: Répartition en plusieurs processus pour le calcul de la force. On a représenté les trois processus d'affichage, de calcul de la force, et d'envoi de cette force au périphérique. On voit aussi les données partagées entre ces processus, et les accès (lecture/écriture) sur chacune de ces données.

Dans notre cas, le champ potentiel scalaire fournit déjà beaucoup d'informations sur la surface.

Par construction de la surface comme iso-potentielle, la détection de collision en un point P se résume à un test entre la valeur de potentiel en P et cette iso-valeur. Dans notre cas, si la valeur de potentiel est plus grande que cette iso-valeur le point P se trouve à l'intérieur de la surface. La force de réaction en P est normale à la surface, et sera donc orientée selon le gradient du champ. R. Avila et L. Sobierajski [AS96, Avi98] déduisent une force en utilisant ce gradient avec la vitesse de déplacement et des fonctions de transfert utilisant la valeur du potentiel pour moduler leurs normes.
De manière similaire, nous déduisons une *force* en un point P à partir d'un terme de viscosité opposé au déplacement et d'un terme prenant en compte la surface.

Terme de surface

Pour prendre en compte la surface, on construit une force dirigée selon le gradient du champ, qui varie avec la valeur du potentiel au point considéré et augmente rapidement lorsque cette valeur dépasse l'iso-valeur (i.e. le pointeur pénètre à l'intérieur de l'objet). Ce qui peut être traduit par :

$$\vec{f_p} = -\frac{\vec{grad}(V_p)}{\|\vec{grad}(V_p)\|} \, min\left(\left(\frac{V_p}{iso}\right)^{\alpha}, maxval\right)$$

où V_p est la valeur du champ potentiel au point p considéré, et α un exposant qui permet de régler la rapidité des variations de la norme de la force autour de l'iso-valeur (i.e. la *pente de la norme*). Cette force est tronquée à *maxVal* pour ne pas dépasser les limites du périphérique (voir Figure 4.4).

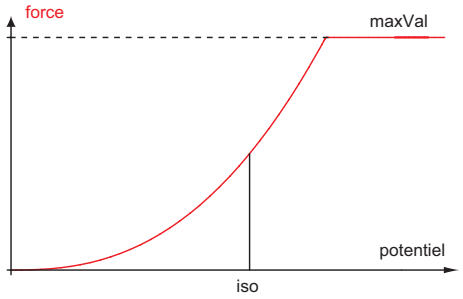


FIG. 4.4: Norme de la force de surface en fonction du potentiel.

On voit que α représente la pente de la fonction norme de la force, par rapport au potentiel, en *iso*. Hors, ce potentiel est lié à la distance à la surface : il vaut *iso* sur la surface, plus à l'intérieur de l'objet, et moins en dehors. Ainsi, α se trouve lié à la distance à la surface, et pourrait donc grossièrement être assimilé à la raideur d'un ressort dans cette zone.

Cela pourrait sembler un moyen élégant de rattacher ce paramètre arbitraire à un paramètre plus *réel* de propriété de surface. En fait, le lien avec la distance n'est pas évident : comme illustré dans la Figure 4.5, la variation du potentiel dépend de l'historique des applications. Pour en donner une idée, remarquons que la variation du potentiel autour de l'iso-valeur, après une application de l'outil ajout de matière (par exemple), est très différente de la variation après un grand nombre d'applications répétées au même endroit (voir Figure 4.5). Cela provient de la manière dont nous prenons en compte les actions d'un outil, par accumulation de ses contributions dans la grille d'échantillonnage.

Ainsi, les variations du potentiel au voisinage de la surface ne sont pas uniformes le long de cette surface. Par conséquent, même si notre force peut s'apparenter à un ressort dans le voisinage de la surface, la raideur théorique du *ressort* équivalent n'est pas constante le long de cette surface

Des études sur la perception de la dureté, par exemple [LPSD96], montrent dans le cas d'un contact avec un plan (tapotements), que la raideur des ressorts utilisée dans les calculs n'est pas liée à

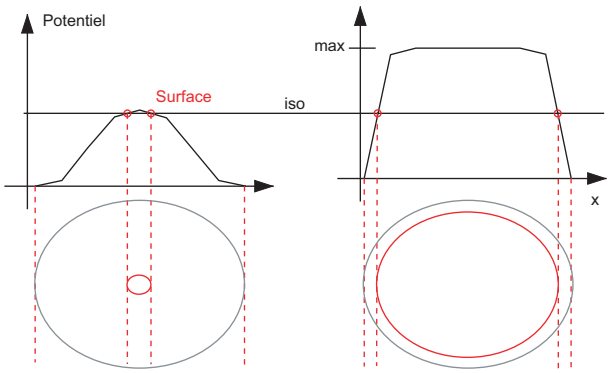


FIG. 4.5: Illustration de la variation de *pente* du champ potentiel au voisinage de l'iso-contour selon le nombre d'applications de l'outil

la dureté perçue. D'après ces expériences, cette dureté perçue serait plus liée au *taux de changement de la force par rapport au taux de déplacement*, soit donc un terme relié au ratio entre la dérivée de la force et la vitesse. Les auteurs pensent que ce sont des fréquences plus hautes qui interviennent dans la perception de la dureté. On retrouvera ce type d'analyse à la fin de cette partie [OHCD00].

Ces études pourraient nous rassurer en montrant que les variations que nous obtenons sur la *raideur théorique* au voisinage de la surface n'ont pas une grande incidence sur la perception. En pratique, nous n'avons pas noté de différence sensible au cours de nos utilisations.

Terme de viscosité

Le terme de viscosité est opposé en direction à la vitesse de déplacement $\vec{v}$. Sa norme est multipliée par la valeur V_p du potentiel au point p considéré, et une constante arbitraire α qui permet d'ajuster son influence avec l'interface :

$$\vec{f}_v = -\alpha V_p \vec{v}$$

L'utilisation du potentiel dans le calcul de cette force permet de ressentir la *densité* de la matière qui s'oppose au déplacement. Ceci est utile comme guide quand on n'est pas en contact avec la surface, comme par exemple lors de l'utilisation d'un outil gomme.

La force renvoyée est la somme pondérée des deux contributions *visqueuse* et *surfacique*. La pondération varie selon que l'outil est appliqué ou pas. En effet, lorsque l'outil n'est pas appliqué, on est en mode exploration de la surface. Le terme de surface prédomine pour assurer le contact / suivi de cette surface. Si on conserve ce mode lors d'un ajout de matière, plus la surface se rapproche du pointeur, plus ce dernier est repoussé, ce qui gêne toute édition précise. L'idée est donc, lorsqu'on modifie la surface, de rendre le terme visqueux prépondérant et diminuer l'influence du terme de surface. La transition entre les deux modes est effectuée progressivement pour éviter les changements brutaux.

Le calcul de cette force ponctuelle se fait quasiment à 1000 Hz, qui est la fréquence requise par le bras PHANTOM Desktop[1]. Toutefois, il apparaît à l'usage des vibrations, que nous avons cherché à atténuer par des techniques de filtrage.

[1] On n'aurait donc, dans ce cas, pas besoin de notre double boucle synchrone / asynchrone (Figure 4.3).

1.3 Traitement des vibrations

Les vibrations proviennent de la présence de l'utilisateur dans la boucle de calcul. Ces problèmes sont similaires à ceux rencontrés en robotique (télémanipulation). Nous allons aborder brièvement quelques analyses pour donner une intuition de l'origine des vibrations avant de détailler notre solution.

Origine des vibrations

La situation la plus simple où les vibrations apparaissent est un plan avec lequel un utilisateur entre en contact. Les vibrations proviennent de problèmes d'échantillonnage et de délais. En effet, les données lues et les forces renvoyées sont discrètes, alors que l'utilisateur qui entre dans la boucle de simulation en s'opposant aux forces calculées à travers le stylet est continu.

Pour décrire la situation informellement, si le pointeur tenu par l'utilisateur est hors du mur virtuel, aucune force n'est renvoyée. Si l'utilisateur pénètre dans le mur, une force dépendant de cette pénétration est renvoyée. Cette situation binaire entraîne des oscillations. En effet, l'utilisateur qui veut maintenir le contact avec le plan compense la force répulsive qui lui est envoyée lorsqu'il a pénétré au delà. Si à la mesure suivante, il a pénétré plus avant dans le mur, une force plus grande sera renvoyée, et les oscillations iront grandissantes.

Ce problème de contact est classiquement contourné en diminuant la raideur du mur et en ajoutant une viscosité artificellement grande [MOyS+90]. Ce type de stabilisation consistant à augmenter artificiellement la viscosité / dissipation du système est assez analogue aux astuces utilisées en animation pour la synthèse d'image. Cette analyse n'est pas sans rappeler les critères de stabilité servant à borner les intervalles de temps entre deux étapes de simulation (exemple du critère de Courant, annexe B de [Deb00]). Dans notre cas toutefois, on ne peut pas diminuer l'intervalle entre deux mesures.
Accessoirement, comme l'utilisateur ressent ces paramètres à travers le stylet, la gamme d'ajustements est assez restreinte et implique des conséquences notables : diminuer la raideur donnera l'impression d'avoir des objets en mousse, et augmenter la friction donnera l'impression de se déplacer dans un milieu visqueux.

La grande difficulté empêchant l'analyse est que l'être humain appartient au système et n'est pas modélisable. M. Minsky évoque dans [MOyS+90] que des modèles du second ordre avec des paramètres dépendant de l'activation des muscles et la posture sont insuffisants, i.e. ne reproduisent pas les résultats expérimentaux. Il est donc impossible de déterminer précisément un pas de temps d'échantillonnage ou une viscosité garantissant la stabilité du système.

Même dans une situation analysable où l'utilisateur est modélisé par un système de second ordre classique (avec des coefficients constants), comme étudiée par R. Gillepsie et M. Cutkosky [GC96], la contrainte sur le pas de temps des mesures peut rendre le système instable pour de grandes raideurs. Dans ce modèle simplifié, l'utilisateur est assimilable à une balle soumise à la pesanteur et rebondissant sur le sol. Une simulation utilisant une réponse du mur de type ressort, avec une raideur donnée, conduit à un système instable pour un pas de temps inadapté (voir Figure 4.6). R. Ellis et al. [ENJ96] proposent une analyse du même problème faisant appel à des considérations énergétiques. Tous deux proposent des solutions stabilisantes par l'utilisation de techniques de prédiction / correction.

La plupart des articles que nous avons consulté traitent uniquement des problèmes liés à la discrétisation dans le temps, mais passent sous silence le fait que l'utilisation d'un calcul ponctuel constitue aussi une discrétisation du problème.
En effet, l'utilisation d'un point face à l'obstacle renvoie une réponse binaire dehors/dedans, qui est

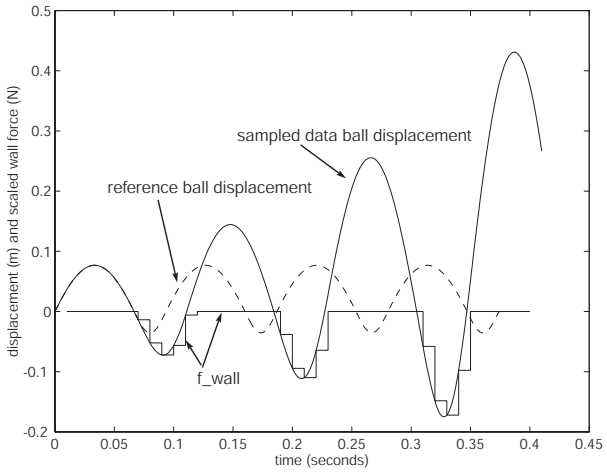


FIG. 4.6: Simulation du rebond d'une balle soumise à la pesanteur rebondissant sur un sol, avec un pas de temps de 0.01 secondes et une réponse du mur (sol) modélisée par un ressort de raideur $K = 500N.M^{-1}$ (image extraite de [GC96]).

ensuite exploitée avec un modèle de ressort pour traduire la raideur du mur. Un calcul plus précis des volumes en jeu contribuerait également à filtrer les réponses calculées. Nous avons essayé ce type d'approche en collectant à chaque itération l'ensemble des points échantillonnant l'objet et couverts par l'outil. Dans notre implémentation monorésolution basée sur les tables de hashage, cette approche de complexité cubique s'est évidemment révélée inexploitable, car fortement dépendante à la fois de la taille de l'outil et la taille de la zone d'interpénétration. La lenteur des calculs augmente beaucoup le temps de mise à jour de la force, rendant le système instable et difficile à utiliser.
Nous n'avons pas essayé ce type d'approche dans notre implémentation multirésolution, où il aurait été sans doute possible de tirer parti de la répartition entre précision et rapidité des calculs.

Nous sommes donc restés sur une solution ponctuelle calculée au centre de l'outil. Remarquons toutefois que notre situation est sensiblement différente du mur virtuel. En effet, la force répulsive que nous calculons ne devient pas nulle immédiatement en quittant la surface, et une contribution répulsive demeure dans la zone où le champ potentiel n'est pas nul. Mais surtout, le problème est plus compliqué car la **forme** du potentiel et de son gradient est **quelconque**. Egalement, le parcours de nos structures ne pouvant garantir un temps de réponse constant, nous avons un **pas d'échantillonnage variable**.

Position amortie

Une alternative pour éviter les oscillations et les instabilités potentielles qu'elles peuvent engendrer, sans faire appel à une friction artificielle est de filtrer ces oscillations. En effet, comme nous avons remarqué que les oscillations étaient des *hautes fréquences*, un filtre passe-bas devrait permettre de les éliminer, et avec elles les instabilités dont elles sont la source.

Nous aurions pu filtrer la force renvoyée, mais comme cette dernière est calculée à partir de la position du pointeur, il nous a semblé plus judicieux de travailler directement sur cette position. Cette position traduit le compromis entre la force exercée par l'utilisateur, et celle imposée par le périphérique.

Pour réaliser ce filtrage, on utilise une *position amortie* exponentiellement q. En supposant que la dernière mesure a eu lieu au temps $(t - \delta t)$, et qu'on lit maintenant la nouvelle position p du pointeur

δt secondes plus tard, on calculera la nouvelle position amortie a_t comme suit :

$$a_t = a_{(t-\delta t)} + \frac{\alpha}{\delta t}\delta p$$

avec $\delta p = p_t - a_{(t-\delta t)}$. La Figure 4.7 représente quelques itérations de ce schéma en considérant un pas de temps constant. Ce filtrage permet bien d'éliminer les vibrations hautes fréquences lors du contact. Toutefois, lorsque l'utilisateur effectue un mouvement de plus grande amplitude, la position

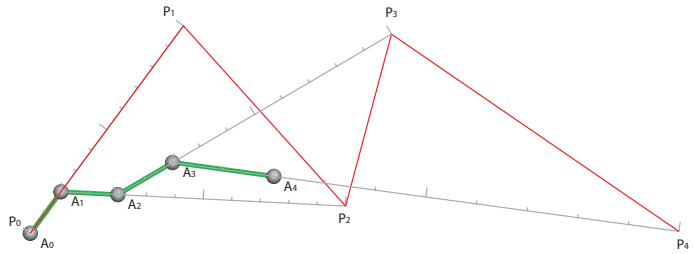


FIG. 4.7: Illustration du calcul de la position amortie, avec un intervalle de temps constant et un ratio $\alpha/\delta t$ valant 0.2.

filtrée introduit un retard non négligeable avant de *rattraper* la position actuelle. Cela entraîne des instabilités / oscillations de plus grande amplitude. On introduit donc une estimation de la *confiance* β en la position amortie, pour en tenir plus ou moins compte dans notre *position estimée* e_t, utilisée pour calculer la force.

$$e_t = \beta a_t + (1-\beta)p_t$$

On réalise une combinaison linéaire entre la position lue des capteurs et la position amortie. La confiance β est estimée à partir de $\|\delta p\|$, la distance entre la position lue et la position amortie. En effet, les oscillations de haute fréquence sont de faible amplitude (au moins si le système n'est pas trop instable !), donc le filtrage mis en place pour les éliminer peut être appliqué. Si la différence de position est *importante*, le filtrage n'est plus utile, même potentielle source d'erreurs. Nous avons ici utilisé :

$$\beta = \frac{\lambda}{\|\delta p\| + \lambda}$$

ce qui donne bien :

- la confiance $\beta \to 1$ quand $\|\delta p\| \to 0$, c'est-à-dire $e_t \to a_t$, la position amortie est plus largement prise en compte si la position lue en est proche (comme c'est le cas pour des petites oscillations haute fréquence).
- la confiance $\beta \to 0$ quand $\|\delta p\|$ devient très *grand*, c'est-à-dire $e_t \to p_t$, la position amortie est ignorée, et on utilise directement la position lue, qui devrait mieux estimer la position virtuelle.

La différence entre la position estimée et la position lue (ou réelle) est bornée. En effet,

$$\|e_t - p_t\| = \beta\|\delta p\| = \frac{\lambda}{\|\delta p\| + \lambda}\|\delta p\| \leq \lambda$$

En pratique, cette différence entre la position estimée et la position réelle n'est pas notable. Ce type de distinction entre la position réelle et la position affichée est de toute façon déjà très répendue dans les techniques de *proxy* utilisées classiquement avec les modèles polygonaux [Avi98], et ne pose pas de problème particulier à l'utilisateur. Notons que cette position estimée n'est pas un *proxy*, ne serait-ce que, par exemple, parce qu'elle n'est pas contrainte à rester en contact avec la surface en cas de pénétration. En particulier, notre technique de calcul de la force n'empêche pas de *passer à travers* un objet.

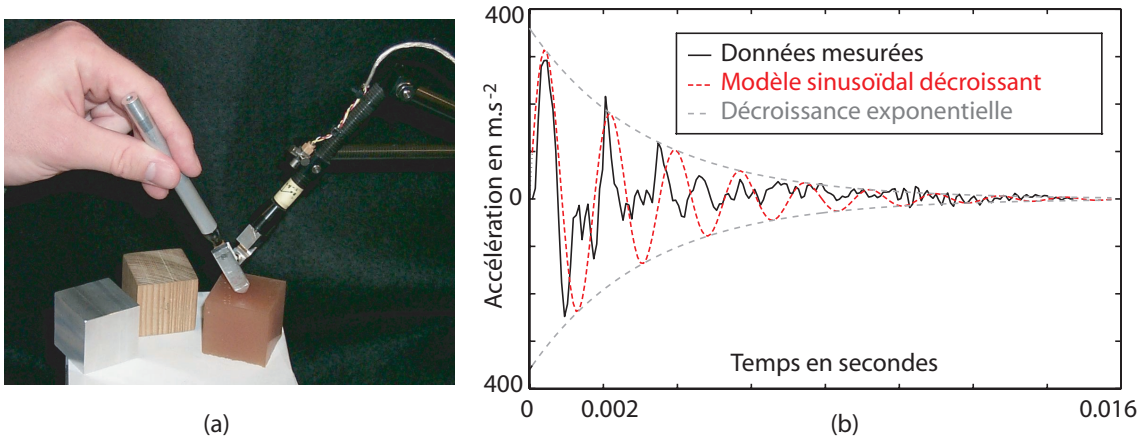


FIG. 4.8: (a) dispositif de mesure des contacts en tapant sur des échantillons de matériaux ; (b) affichage du modèle sinusoïdal décroissant avec les données mesurés avec du bois (images extraites de [OHCD00]).

Nous n'avons pas analysé plus finement le filtrage effectué sur la position. Une telle analyse pourrait éventuellement bénéficier des travaux plus récents sur les mesures des vibrations et leurs atténuations effectuées sur des contacts avec des matériaux réels. Par exemple A. Okamura et al. [OHCD00] présentent un dispositif pour mesurer les oscillations obtenues en tapant avec un stylet sur des matériaux réels. Ils utilisent un accéléromètre pour collecter des données à 25kHz, et montrent l'importance de la restitution de ces oscillations pour la perception des surfaces.
Il n'est donc pas souhaitable de complètement supprimer ces oscillations, et une caractérisation plus poussée du filtre pourrait permettre d'identifier les paramètres pour reproduire les atténuations mesurées.

Bilan

La force obtenue précédemment nous est apparue satisfaisante, car elle permet une *bonne* sensation de la surface. C'est-à-dire que l'on peut confortablement explorer la forme en suivant la surface, et les manipulations d'outils sont facilitées par le guide procuré avec la viscosité du potentiel. Cela s'avère utile à la perception et à la compréhension de la surface, en particulier en mode exploration, où l'on suit la surface en maintenant le contact.
La force calculée n'est pas précise quantitativement et elle ne prend pas en compte les éventuelles collisions entre l'outil et la surface. Hormis dans des situations particulières où l'intersection devrait empêcher le passage de l'outil, comme par exemple la traversée d'un anneau, cela n'est pas remarquable. Dans une situation comme celle évoquée avec l'anneau, l'information visuelle indique que l'outil ne peut passer, alors que les forces calculées au point centre de l'outil peuvent, dans un cas extrême, être nulles. Ces incohérences temporaires, comme l'absence de *proxy* pour empêcher de transpercer la surface, et donc la possibilité de passer à travers si la force exercée par l'utilisateur est trop grande, ne sont pas apparues perturbantes à l'usage.

2 Opérations d'édition

A l'usage le périphérique à retour d'effort de type stylet constitue une aide précieuse à la construction / édition de l'objet sculpté. D'autres fonctionnalités comme le copier / coller deviennent très rapidement essentielles dans ce processus. En effet, la possibilité de pouvoir sélectionner le bras gauche d'un personnage en cours de construction, pour pouvoir initaliser le droit, permet de gagner du temps

et d'économiser ses efforts. De même, si la jambe sur laquelle on vient de terminer le pied est mal positionnée, être obligé de tout effacer et recommencer pour obtenir la bonne position n'est pas envisageable.

Toutefois, la sélection n'est pas facile avec la représentation volumique que nous utilisons. En effet, on pourrait imaginer de peindre la surface à sélectionner, mais il faudrait ensuite être capable de remonter au potentiel contenant cette surface. Cette opération n'est pas immédiate, car il est nécessaire d'identifier pour chaque échantillon du potentiel la surface à laquelle il contribue (le plus ?). On pourrait imaginer d'utiliser des techniques de propagation de front à partir de la surface sélectionnée. Quoiqu'il en soit, ces recherches seraient forcément coûteuses, car elles demandent des consultations de voisinage pour chaque échantillon examiné.

De la même manière, en imaginant qu'on arrive à retrouver ce potentiel, et donc à réaliser la sélection, l'opération de découpage ne serait pas facile non plus. Il n'est pas envisageable de découper directement la sélection. Dans ce cas, il apparaîtrait des discontinuités dans le champ potentiel scalaire, et le raccord de ce potentiel découpé souffrirait des mêmes discontinuités, et à fortiori les surfaces correspondantes.

Par ailleurs, comme l'utilisateur ne perçoit que la surface, il n'est pas non plus concevable de lui faire sélectionner explicitement le potentiel qui l'intéresse : il ne devrait même pas avoir connaissance de son existence. De plus, cela constituerait une manipulation directe de la représentation que nous cherchons justement à éviter ici.

Une solution élégante à ces problèmes, introduisant la notion de calque de sélection, a été apportée par Renaud Blanch lors de son stage.

2.1 Notion de calque de sélection

Lorsque l'utilisateur entre en mode sélection, l'outil sert à construire un nouvel objet, similaire à la sculpture elle-même, et affiché en transparence (voir Figure 4.9.(a)). Ce nouvel objet est construit avec les mêmes outils que ceux utilisés pour la sculpture. Par analogie avec les logiciels $2D$ de retouche d'images, cet objet est appelé calque de sélection.

En fait, une structure identique à celle représentant la sculpture, c'est-à-dire l'ensemble des trois structures de sommets, arêtes et cubes (*CornerTree*, *EdgeTree* et *CubeTree*) de notre première implémentation, sert à *sculpter* un calque / volume de sélection. Cette nouvelle structure d'échantillonnage de champ potentiel est alignée avec celle contenant la sculpture. On évite ainsi les problèmes de ré-échantillonnage. Le potentiel stocké dans le calque peut être vu, non plus comme une *densité de matière*, mais plutôt comme un *degré de sélection*.

Le calcul de la sélection consiste à construire une troisième structure stockant le potentiel résultant du potentiel de la sculpture initiale, en tenant compte de celui du calque. Concrètement, on parcourt tous les sommets du calque et le potentiel stocké dans la sélection est le plus petit des deux potentiels sculpture et calque.

L'iso-surface de la sélection ainsi contruite est l'intersection de la surface sculpture et la surface calque. Le découpage **progressif** obtenu grâce au potentiel du calque réalise un lissage dans les zones de sélection où le potentiel sculpture dépasse l'iso-valeur, i.e. découpe à l'intérieur de l'objet. En effet, dans ces zones, c'est le potentiel (plus faible) du calque qui est utilisé, et assure une décroissance progressive vers un potentiel nul (à la limite du calque). Cette décroissance au-delà de la surface est utile pour éviter les problèmes de bruit sur la surface, et aussi comme on l'a vu dans la partie précédente,

pour le rendu haptique.

Du point de vue utilisateur, les potentiels en jeu sont invisibles, et la sélection obtenue comprend bien les parties de la surface sculpture délimitées par la surface calque, ce qui correspond au comportement attendu.

2.2 Couper copier coller

Une fois le potentiel de sélection calculé, cette sélection peut être utilisée comme un outil classique.

(a) (b) (c)

FIG. 4.9: Calque de sélection : (a) La sculpture et le calque de sélection englobant une jambe ; (b) La jambe coupée et la sélection affectée comme outil ; (c) La jambe collée avec une position différente.

Il est donc possible de réaliser un collage par simple application de cet outil. La découpe est réalisée en soustrayant la sélection immédiatement après sa construction (voir Figure 4.9 (a) et (b)). Le *lissage* induit par le potentiel du calque permet d'éviter les problèmes de crénelage dans les régions ou la surface calque intersecte la surface sculptée.

FIG. 4.10: Exemple de sculpture réalisé en quelques heures par Renaud Blanch lors de son stage, avec le premier prototype et un PHANTOM Desktop.

Ces opérations permettent de modifier facilement une sculpture en cours de construction (voir Figure 4.10).

C'est ici que s'arrêtent nos contributions. Les parties qui suivent sont plus des discussions ou des compléments sur des aspects non couverts dans les articles consacrés au deux prototypes.
Il nous est apparu toutefois plus logique de faire figurer ces *perspectives* ici plutôt qu'en conclusion du document.

2.3 Techniques de déformation

Avec les opérations de couper / coller exposées précédemment, il est possible d'ajuster la sculpture relativement facilement, comme par exemple la jambe de la Figure 4.9. Toutefois, si l'on souhaite tordre un élément au lieu de le déplacer, il est nécessaire de disposer d'opérations de déformations plus riches.

Notre choix de privilégier des périphériques d'entrée $3D$ et un affichage stéréoscopique interdit à priori toute interaction utilisant des courbes dessinées sur l'écran, comme proposé dans Teddy par exemple.

On a vu dans la première partie de ce document que les FFD constituaient un moyen assez souple pour déformer globalement un objet. Nous avions envisagé de déformer le volume de sculpture par ces techniques. Nous avons échangé quelques réflexions sur ce sujet avec Laurent Grisoni, alors en thèse au LaBRI.
La surface triangulée de la sculpture aurait été *gelée*, et utilisée pour la visualisation interactive des déformations. La validation d'une déformation aurait ensuite entraîné le ré-échantillonnage de tout le champ potentiel ainsi déformé, opération à priori non interactive.

En effet, cette opération de ré-échantillonnage est plus complexe qu'il n'y paraît. Il faut d'une part retrouver pour chaque point de la grille d'échantillonnage sa position dans l'espace de paramètres du volume FFD **déformé**. Le problème est inversé par rapport à l'utilisation classique des FFD avec des surfaces *explicites*. On rencontre le même genre de problème lorsqu'on effectue un lancer de rayon sur ces volumes déformés [Bar86]. D'autre part, il aurait fallu gérer les problèmes d'aliassage lors de ce ré-échantillonnage.

A l'époque de ces réflexions, nous ne disposions pas encore de l'implémentation multi-résolution, et les limitations imposées par la résolution fixe de la grille destination nous ont fait repousser ces considérations à cette prochaine implémentation. Ce délai a été fatal à ces réflexions.
Nous avions par ailleurs plusieurs problèmes liés sans vraiment de pistes pour les résoudre, comme la spécification de ce volume FFD et de ses déformations. Comment déduire, à partir de la surface, le volume à mettre en place, sa résolution ? Et ensuite, comment le déformer sans passer par la manipulation des points de contrôles (inspiration de [HHK92] ?) ?

Une autre voie pour spécifier ces déformations, et qu'il serait intéressant d'explorer utilise un calcul d'*axe médian*, comme proposé par J. Bloomenthal et C. Lim dans [BL99] (voir Figure 4.11). D'autres techniques d'extraction de courbes *structurantes* existent, comme présenté par A. Verroust et F. Lazarus [VL00]. L'intérêt de ces techniques est d'arriver à déduire automatiquement un axe à partir de la surface sélectionnée. Cette paramétrisation automatique de la surface peut aussi être utilisée dans notre cas pour paramétrer la représentation de potentiel associée.

Notons qu'à partir de cet axe, il est possible de manipuler la surface (et avec elle, son champ potentiel). L'utilisation de l'axe, qui parfois simplifie l'édition, n'est toutefois pas obligatoire. Une piste pour agir sur cet axe à partir de la surface est proposée par J. Corso et al. dans [CCO02]. Ils étudient une technique pour déformer interactivement une surface décrite par son axe médian (voir le principe dans la Figure 4.12.(a)). L'article s'inscrit déjà dans un cadre d'interaction avec un bras

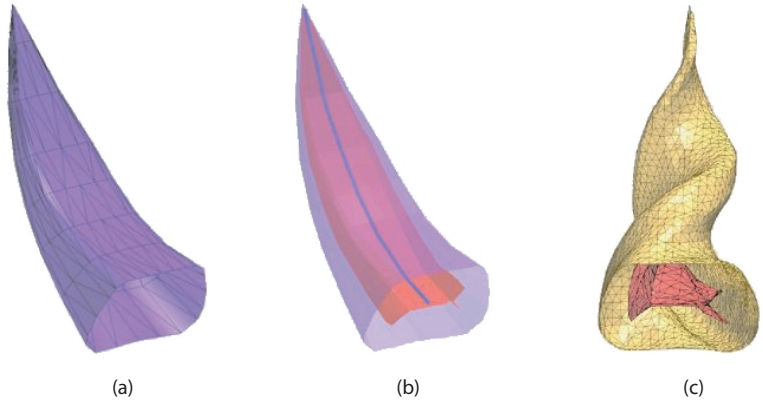


FIG. 4.11: Illustration du principe d'édition par axe médian : (a) une surface polygonale ; (b) calcul de l'axe médian associé ; (c) déformation (images extraites du site de J. Bloomenthal (`www.unchainedgeometry.com/jbloom`)).

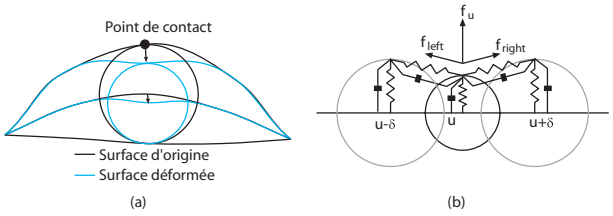


FIG. 4.12: (a) la déformation appliquée à est répercutée sur l'axe médian. (b) illustre le modèle masse-ressorts amortis mis en place pour calculer les forces en utilisant l'axe médian (schémas extraits de [CCO02]).

articulé à retours d'effort, et propose une solution utilisant un modèle masses-ressort amortis, avec un ressort relié à l'axe médian pour calculer la composante normale, et quatre ressorts liés à la surface pour gérer les torsion (Figure 4.12.(b)).

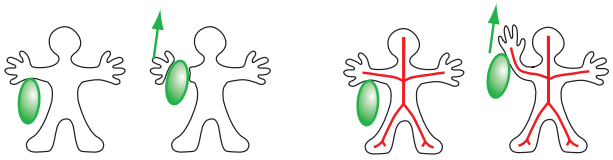


FIG. 4.13: Notion de déformation à partir de la surface, exploitant la représentation axiale associée.

L'avantage de ces techniques, construisant un axe associé à la surface, par rapport à une déformation directe de la surface, est qu'elles ouvrent la voie à des déformations plus globales (voir Figure 4.13).

3 Remarques sur les solutions proposées

Les implémentations que nous avons proposées sont des implémentations de tests, et ne sont pas optimales.

3.1 Stockage du champ potentiel

En particulier, nous sotckons des informations redondantes, comme la position qui pourrait facilement n'être stockée qu'au noeud racine, est maintenue lors du parcours de la structure, cette structure constituant une sorte d'itérateur, comme utilisé dans les STL.

Nous n'avons pas cherché non plus à optimiser le stockage des autres informations, comme la couleurs ou les gradients du champs, qui pourraient bénéficier comme dans [AS96] de codage par des tables par exemple.

3.2 Représentation de la surface

Une question qui est souvent revenue lors des présentations de ces travaux, concerne le stockage de la surface. Dans nos deux implémentations, la surface n'est jamais stockée, c'est-à-dire qu'elle ne bénéficie pas d'une structure dédiée pour décrire ses faces et ses arêtes.

- Lors de la mise-à-jour des valeurs de potentiel aux coins d'une cellule, la comparaison de ces 8 valeurs avec l'iso-valeur donne directement un index de configuration dans la table de Marching Cube. Cet index sert à savoir, via une *table d'arêtes*, quelles sont les arêtes où ces intersections se produisent. Ces dernières sont donc calculées par interpolation (en fonction du potentiel, le gradient et les attributs comme la couleur, sont interpolés comme cela est fait classiquement [BBB+97]), et stockées.
- Lors de l'affichage, cet index permet via une *table de cubes*, de savoir combien de triangles la surface génère dans la cellule et quelles arêtes ils relient.

Cette représentation de la surface ne permet pas, en particulier, de gérer les problèmes de déchirures évoqués dans le second prototype. Au cas où deux cellules adjacentes ont des résolutions différentes, comme leur affichage est effectué de manière complètement autonome, rien ne garantit que les triangles obtenus sont jointifs.
Nous avons choisi explicitement de ne pas gérer ces déchirures lors de la construction interactive, où de toute façon la surface est en continuelle évolution.
Cette gestion des déchirures nécessite la consultation des cellules voisines, et dans l'affichage guidant la complexité que nous avons proposé, les voisins affichés ne sont potentiellement pas les mêmes d'une image à l'autre.

Nous avons laissé cette *couture* des déchirures pour une éventuelle phase d'exportation, où la surface pourrait aussi bénéficier de simplifications [GH98]. Une autre alternative intéressante dans ce cadre d'exportation pourrait être, comme proposé dans [Sou01], de rejouer l'intégralité des commandes d'édition à des résolutions plus fines.

Nous n'avons pas exploré non plus des techniques de rendu alternatives, comme par exemple le rendu par points proposé par *QSplat* [RL00], et qui pourrait également exploiter judicieusement la représentation par octree dont nous disposons pour stocker le champ potentiel.

4 Champ potentiel

Une limitation des représentations que nous utilisons ici, est inhérente au stockage du potentiel, qui reconstruit par interpolation trilinéaire le potentiel à l'intérieur des cellules. Cela pose problème pour l'échantillonnage des discontinuités du champ potentiel. Notre solution ici, a consisté à diminuer la résolution jusqu'à un seuil limite, relié à la taille de l'outil utilisé.

Des alternatives intéressantes, consistant à spécialiser certaines cellules pour stocker une discontinuité d'arête ou de coin, comme proposé par exemple dans [BN90] avec les *Extended Octree*, devraient lever ces limitations (et incidemment réduire la profondeur de la hiérarchie). Nous n'avons toutefois pas exploré concrètement (i.e. implémenté) ce type d'approche.

Des extensions intermédiaires, notament sur l'algorithme de Marching Cubes, comme exposé dans [KBSS01] (voir Figure 4.14) auraient également pu contribuer à améliorer la gestion des dis-

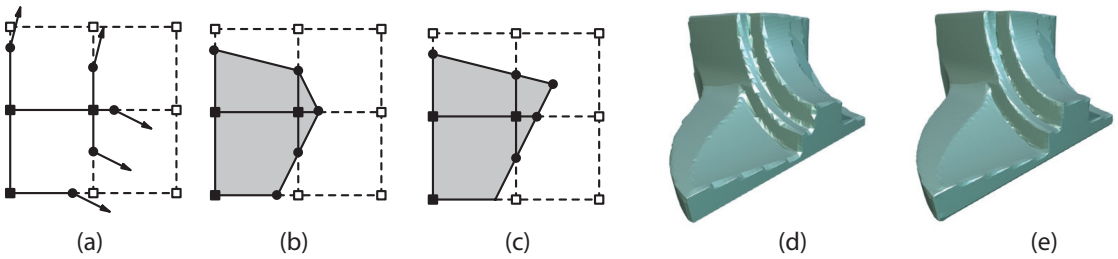


FIG. 4.14: (a) grille d'échantillonnage, avec en noir les points où le potentiel dépasse l'iso-valeur. On voit sur les arêtes les intersections calculées et les gradients interpolés. (b) l'iso-surface extraite avec un Marching-Cubes classique. (c) l'iso-surface extraite avec un Marching-Cubes étendu (figures extraites de [JLSW02]). (d) et (e) comparent les surfaces extraites du même champ *potentiel* avec un Marching Cubes classique et étendu (images extraites de [KBSS01]).

continuités. L'idée de ce *Marching Cubes étendu* est d'exploiter les informations du gradient pour extraire une représentation plus fine de la surface par cellule (voir la Figure 4.14, détournée de l'article [KBSS01]). L. Kobbelt et al. dans [KBSS01] travaillent sur un champ de distance signé, analogue aux champs de distance présentés dans [FPRJ00] que nous avons abordé dans le première partie du document, mais l'extraction de la surface avec un champ potentiel est identique.

Ce dernier article expose également la notion de champ de distance étendu, ou *directed distance field*, qui consiste à stocker avec la distance la direction vers la surface la plus proche. Ce type de représentation pourrait avantageusement remplacer le champ potentiel que nous stockons, en particulier pour éviter les problèmes de variations avec l'historique de construction le long de la surface, comme noté à la Figure 4.5.

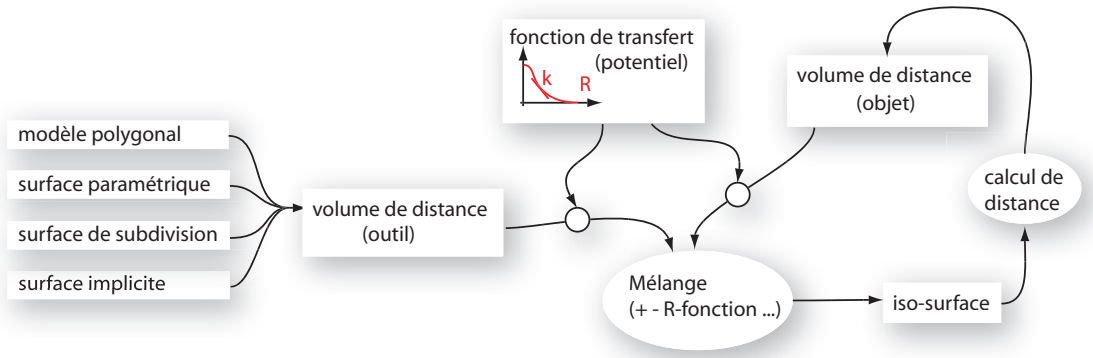


FIG. 4.15: Schéma de principe exploitant des champs de distance.

Nous avions imaginé, au cours de nos réflexions sur le sujet, utiliser des volumes de distance pour représenter l'outil et l'objet (voir Figure 4.15). L'intérêt d'une telle représentation serait d'offrir plus de souplesse, en permettant, par exemple, de choisir au moment du mélange des deux champs (application de l'outil), les paramètres de mélange[2] de chacun, ainsi que le type d'opération de mélange effectué. La mise-à-jour du champ de distance objet pourrait être conduite à partir de l'iso-surface localement extraite (la surface modifiée est bornée par les régions d'influences des primitives implicites).

Nous avons vu toutefois dans la première partie du document que par rapport à la mise-à-jour du potentiel, la mise-à-jour de la distance n'est pas locale, et il pourrait être intéressant de creuser dans

[2] *raideur k* et *extension R*, par exemple, de la fonction potentiel utilisée pour convertir les volumes de distance en primitives implicites.

cette direction pour définir une distance *étendue* qui deviendrait moins précise avec l'éloignement de la surface. Ceci rejoint les fondements des niveaux de détails et de la simplification géométrique :

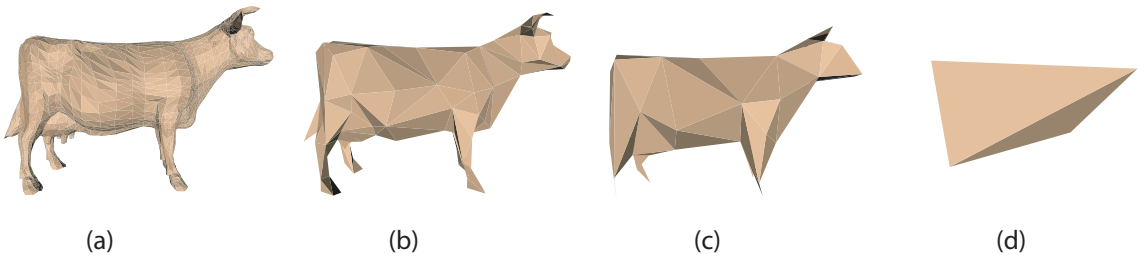


FIG. 4.16: Exemple de simplification du maillage d'une vache : (a) 5804 faces, (b) 300 faces, (c) 100 faces et (d) 4 faces (images extraites de [Gar99], pages 103 et 104).

plus l'objet est éloigné, moins il occupe de surface sur l'écran, moins sa représentation a besoin d'être précise. Dans la situation limite, la vache de la Figure 4.16 pourrait se réduire à un point.

Conclusion

Nous avons présenté dans ce document deux approches basées sur des surfaces implicites discrètes, c'est-à-dire échantillonnant un champ potentiel scalaire, qui peut alors être considéré comme une pseudo-densité de matière.

Notre premier prototype était plus destiné à se convaincre de la viabilité de l'approche tant en terme d'interactivité que d'interface, qu'a apporter des réponses originales. Nous en avons profité toutefois pour utiliser des structures dynamiques de type arbre binaire équilibré, et tables de hashage pour éviter les limitations imposées par les approches précédents notre implémentation, notament sur l'extension spatiale du volume échantillonné. Comme notre but était de rendre l'interaction la plus directe et libre possible, il était dommage d'imposer de travailler uniquement dans une grille fixée au départ.

Cette implémentation de test nous a permis de valider les concepts d'outils et d'interactions que nous avions envisagés : à savoir une visualisation stéréoscopique, et une manipulation de l'outil et de la scène via une souris 3*D* Spacemouse. Nous y avons aussi adapté les formulations de collisions *géométriques* développées dans le contexte des surfaces implicites *continues*. Toutefois, cet aspect ne s'est pas révélé convainquant à l'utilisation, partiellement sans doute à cause de la lenteur induite par l'extension de l'outil à la zone de déformations.

C'est également dans cette implémentation que nous avons, avec Renaud Blanch, interfacé le dispositif à retours d'effort PHANTOM Desktop. Cette solution de retours d'effort *qualitatif* essentiellement guidé par l'intuition, nous a conduit par ses oscillations et instabilités sur les chemins des lois de commandes, de la robotique, où nous ne nous sommes pas aventuré très loin, car pas vraiment à l'aise.

Durant son stage, Renaud a également développé l'idée originale de calque de sélection, qui résoud avec élégance les problèmes de sélection de potentiels associés à partir des surfaces affichées.

Notre second prototype nous a permis de valider une représentation multirésolution et adaptative d'échantillonnage du potentiel. Nous avons proposé de subdiviser chaque cellule en N^3, au lieu des 2^3 usuels, pour accélérer la division et réduire la profondeur de l'octree. Toutefois, nous n'avons pas pu vraiment analyser l'impact (ou l'intérêt) de ce facteur. L'octree, complètement dynamique n'a pas de racine figée, i.e. on peut agrandir arbitrairement son extension, ni de profondeur maximum. Cela nous a conduit à étudier des critères de subdivision, notament pour la stopper le long des discontinuités, et surtout pour garantir un échantillonnage *suffisant*. Nous avons pour cela proposé d'utiliser les

informations comme la taille de l'outil pour guider ces critères.

Dans cette maquette, nous avons pu aussi garantir une mise à jour interactive du champ et de la visualisation en exploitant les parcours horizontaux, par niveau de profondeur, de l'octree. Nous avons mis en place des technique pour contrôler la complexité d'affichage et maintenir l'interactivité lors des manipulations.

La gestion proposée par file d'attente pour le traitement des actions des outils dans un processus séparé nous apparaît également prometteuse pour la gestion de plusieurs outils simultanés, éventuellement dirigés depuis des machines distantes.

Nous avons finalement abordé dans la dernière partie de ce document quelques faiblesses identifiées de notre approche, notament pour la gestion des discontinuités de la surface, et proposé quelques pistes qui nous semblaient prometteuses pour y répondre.

Il nous semble intéressant également dans les proglongements de ces travaux, de les évaluer dans d'autres contextes d'utilisation. Nos développements nous ont permis d'aborder des visualisations sur écran, ou avec un visiocasque immersif, ou dans une salle de réalité virtuelle. La manipulation, mono-utilisateur, était assurée par une Spacemouse pour la scène et un bras articulé à retour d'effort pour l'outil. Des extensions impliquant plusieurs utilisateurs ont été entamés par L. Grisoni [GDC+02] à partir du premier prototype. D'autres interfaces, comme un *Responsive Workbench* ou bien le *StudierStube* évoqués dans la première partie constituent aussi des plateformes intéressantes.

TABLE DES MATIÈRES

TABLE DES FIGURES

BIBLIOGRAPHIE

[ABM+99] R.S. Avila, C. Basdogan, T. Massie, D. Staples, D. Ruspini, K. Salisbury, and R. Taylor. Haptics : from basic principles to advanced applications. *Computer Graphics*, August 1999. SIGGRAPH'99 Course Notes #38. (référencé page 118)

[AS96] R.S. Avila and L.M. Sobierajski. A haptic interaction method for volume visualization. *Computer Graphics*, pages 197–204, October 1996. Proceedings of Visualization'96. (référencé pages 60, 63, 78, 87, 87, 91, 92, 95, 97, 101, 110, 120, 130, 142, 143)

[Avi98] R.S. Avila. Volume haptics. *Computer Graphics*, pages 103–123, July 1998. SIGGRAPH'98 Course Notes #01. (référencé pages 97, 118, 120, 124, 145)

[AYS98] N. Asamura, N. Yokoyama, and H. Shinoda. Selectively stimulating skin receptors for tactile display. *IEEE Computer Graphics and Applications*, pages 32–37, November 1998. (référencé pages 74, 143)

[Bær98] Andreas Bærentzen. Octree-based volume sculpting. *Presented at IEEE Visualization '98*, 1998. `www.gk.dtu.dk/Andreas/publications.html`. (référencé pages 62, 101, 110)

[Bar84] A.-H. Barr. Global and local deformations of solid primitives. *Computer Graphics*, 18(3) :21–30, July 1984. Proceedings of SIGGRAPH'84. (référencé pages 11, 21, 54)

[Bar86] A.-H. Barr. Ray tracing deformed surfaces. *Computer Graphics*, 20(4) :287–296, August 1986. Proceedings of SIGGRAPH'86 (Dallas). (référencé page 128)

[BB89] B.A. Barsky and J.C. Beatty. A technique for the direct manipulation of spline curves. In *Graphics Interface'89*, pages 33–39, 1989. (référencé pages 17, 18, 138)

[BBB+97] Jules Bloomenthal, Chandrajit Bajaj, Jim Blinn, Marie-Paule Cani, Alyn Rockwood, Brian Wyvill, and Geoff Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufman, 1997. (référencé pages 87, 100, 130)

[BDHO92] J. Butterworth, A. Davidson, S. Hench, and T.M. Olano. 3dm : A three dimensional modeler using a head-mounted display. *Computer Graphics*, 25(2) :135–138, 1992. Proceedings of the Symposium on Interactive 3D Graphics. (référencé pages 80, 82)

[BFK84] Wolfgang Böhm, Gerald Farin, and Jurgen Kahmann. A survey of curve and surface methods in cagd. *Computer Aided Geometric Design*, 1(4) :1–60, 1984. North Holland. (référencé page 15)

[BH98] J. Bloomenthal and J.C. Hart. The implicit site. `implicit.eecs.wsu.edu`, last updated May 19, 1998. (référencé page 52)

[BL95] J.R. Bill and S.K. Lodha. Sculpting polygonal models using virtual tools. In *Graphics Interface'95*, pages 272–278, Quebec, Canada, May 1995. (référencé pages 11, 12, 86, 137)

[BL99] Jules Bloomenthal and Chek Lim. Skeletal methods of shape manipulation. *Proceedings of Shape Modeling International '99*, March 1999. (référencé page 128)

[Bla00] Renaud Blanch. Virtual sculpture with haptic feedback, September 2000. Training period report from Supelec (internal report, in french). (référencé page 115)

[Bli82] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, pages 235–256, July 1982. (référencé page 49)

[Blo87] J. Bloomenthal. Polygonization of implicit surfaces. *Xerox Technical Report CSL-87-2*, pages 1–19, May 1987. (référencé pages 52, 88)

[Blo88] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5 :341–355, 1988. (référencé pages 52, 88)

[Blo94] Jules Bloomenthal. An implicit surface polygonizer. In Paul Heckbert, editor, *Graphics Gems IV*, pages 324–349. Academic Press, Boston, 1994. (référencé pages 52, 88)

[Blo95] J. Bloomenthal. Skeletal design of natural forms. *phD thesis, University Calgary,Alberta*, January 95. (référencé page 51)

[BN90] P. Brunet and I. Navazo. Solid representation and operation using extended octrees. *ACM Transactions on Graphics*, 9(2) :170–197, April 1990. ISSN 0730-0301. (référencé page 130)

[Bor94] P. Borrel. Simple constrained deformations for geometric modeling and interactive design. *ACM Transactions on Graphics*, 13(2) :137–155, April 1994. (référencé page 24)

[Bou97] P. Bourke. Web-page on modelling with implicit surfaces. `www.swin.edu.au/astronomy/pbourke/modelling/implicitsurf`, July 1997. (référencé pages 49, 65)

[BS91] J. Bloomenthal and K. Shoemake. Convolution surfaces. *Computer Graphics*, 25(4) :251–256, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991). (référencé page 50)

[BS95] C. Blanc and C. Schlick. Extended field functions for soft objects. *Proceedings of Implicit Surfaces'95*, pages 21–32, April 1995. (référencé pages 49, 141)

[BW90] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2) :109–116, March 1990. (référencé pages 49, 50, 51, 52, 141)

[Can93] M.-P. Cani. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics*, 27 :313–320, August 1993. Proceedings of SIGGRAPH'93 (Anaheim, CA), (Published under the name Marie-Paule Gascuel). (référencé pages 49, 54, 55, 141)

[CCO02] J. Corso, J. Chhunagi, and A. Okamura. Interactive haptic rendering of deformable surfaces based on the medial axis transform. *Proceedings of Eurohaptics 2002*, July 2002. (référencé pages 128, 129, 146)

[CD97] M.-P. Cani and M. Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1) :39–50, March 1997. Published under the name Marie-Paule Cani-Gascuel. (référencé page 92)

[CG91] G. Celniker and D. Gossard. Deformable curve and surface finite-element for free-form shape design. *Computer Graphics*, 25(4) :257–266, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas). (référencé pages 30, 31)

[CLE98] Y.-H. Chai, G.R. Luecke, and J.C. Edwards. Virtual clay modeling using the isu exoskeleton. *Proceedings of VRAIS'98*, March 1998. (référencé page 86)

[CN93] T.J. Cullip and U. Neumann. Accelerating volume reconstruction with 3d texture hardware. *Technical Report 93-027*, 1993. University of North Carolina, Chapel Hill N.C. (référencé page 52)

[CNSD93] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality : The design and implementation of the cave. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 135–142, Anaheim, California, August 1993. ISBN 0-201-58889-7. (référencé page 67)

[CON99] B. Cabral, M. Olano, and P. Nemec. Reflection space image based rendering. *Computer Graphics*, pages 165–170, August 1999. Proceedings of SIGGRAPH'99 (Los Angeles). (référencé page 94)

[Coq90] S. Coquillart. Extended free-form deformation : a sculpturing tool for 3d geometric modeling. *Computer Graphics*, 24(4) :187–193, August 1990. Proceedings of SIGGRAPH'90 (Dallas). (référencé pages 21, 22, 138)

[CP97] Frédéric Cazals and Claude Puech. Bucket-like space partitioning data structures with applications to ray-tracing. In *13th ACM Symposium on Computational Geometry*, 1997. http ://www-imagis.imag.fr/Publications/1997/CP97. (référencé page 103)

[Cre99] B. Crespin. Implicit free-form deformations. *Proceedings of Implicit Surface 99*, pages 17–23, 1999. (référencé pages 25, 138)

[CW92] G. Celniker and W. Welch. Linear constraints for deformableB-spline surfaces. *Computer Graphics*, pages 165–170, 1992. Proceedings, Symposium on Interactive 3D graphics. (référencé page 31)

[DDCB01] 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*, Annual Conference Series. ACM Press / ACM SIGGRAPH, Aug 2001. Proceedings of SIGGRAPH'01. (référencé page 30)

[Deb00] Gilles Debunne. *Animation multirésolution d'objets déformables en temps-réel, Application à la simulation chirurgicale*. PhD thesis, Institut National Polytechnique de Grenoble, Dec 2000. (référencé pages 30, 122)

[Dec96] P. Decaudin. *Modélisation de Formes 3D pour la Synthèse d'Images - Rendu de scènes 3D imitant le style dessin animé*. PhD thesis, Université Technologique de Compiègne, December 1996. thèse de doctorat. (référencé page 25)

[DKT98] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. *Proceedings of SIGGRAPH 98*, pages 85–94, July 1998. ISBN 0-89791-999-8. (référencé pages 38, 39, 40, 43, 139, 140)

[DTC96] Mathieu Desbrun, Nicolas Tsingos, and Marie-Paule Cani. Adaptive sampling of implicit surfaces for interactive modelling and animation. *Computer Graphics Forum*, 15(5) :319–325, December 1996. ISSN 0167-7055. (référencé pages 53, 54, 141)

[EDD+95] Matthias Eck, Tony D. DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 173–182, Los Angeles, California, August 1995. ACM SIGGRAPH / Addison Wesley. ISBN 0-201-84776-0. (référencé pages 41, 140)

[ENJ96] R. E. Ellis, N.Sarkar, and M. A. Jenkinsy. Numerical methods for the force reflection of contact, November 1996. Proceedings of the 1996 ASME International Mechanical Engineering Congress and Exhibition, DSC-Vol. 58. (référencé page 122)

[FB88] David R. Forsey and Richard H. Bartels. Hierarchical B-spline refinement. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22(4) :205–212, August 1988. Held in Atlanta, Georgia. (référencé pages 15, 16, 17, 18, 31, 86, 137, 138)

[FCG99] Eric Ferley, Marie-Paule Cani, and Jean-Dominique Gascuel. Virtual sculpture (short paper), September 1999. Proceedings of Eurographics '99. (référencé page 101)

[FCG00] Eric Ferley, Marie-Paule Cani, and Jean-Dominique Gascuel. Practical volumetric sculpting. *the Visual Computer*, 16(8) :469–480, December 2000. A preliminary version of this paper appeared in Implicit Surfaces'99, Bordeaux, France, sept 1999. (référencé pages 101, 103, 104, 110, 115)

[Fow92] B. Fowler. Geometric manipulation of tensor product surfaces. *Computer Graphics*, pages 101–108, 1992. (référencé pages 18, 19, 79, 138)

[FPRJ00] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields : A general representation of shape for computer graphics. *Proceedings of SIGGRAPH 2000*, pages 249–254, July 2000. ISBN 1-58113-208-5. (référencé pages 63, 101, 131)

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics, Principles and Practice, Second Edition*. Addison-Wesley, 1990. Held in Reading, Massachusetts. (référencé pages 10, 15)

[FW98] David Forsey and David Wong. Multiresolution surface reconstruction for hierarchical b-splines. *Graphics Interface '98*, pages 57–64, June 1998. (référencé page 17)

[Gal97] E. Galin. Métamorphose et visualisation de blobs à squelettes. *Thèse de doctorat, Université Lyon I*, July 1997. (référencé pages 49, 141)

[Gar99] Michael Garland. Quadric-based polygonal surface simplifcation. *CMU-CS-99-105*, May 1999. (référencé pages 132, 146)

[GC96] R. Brent Gillespie and Mark R. Cutkosky. Stable user-specific haptic rendering of the virtual wall, November 1996. Proceedings of the 1996 ASME International Mechanical Engineering Congress and Exhibition, DSC-Vol. 58. (référencé pages 122, 123, 146)

[GD99] James Gain and Neil Dodgson. Adaptive refinement and decimation under free-form deformation. *Eurographics UK'99*, April 1999. (référencé page 25)

[GDC+02] L. Grisoni, S. Degrande, C. Chaillou, E. Ferley M.-P. Cani, and J.-D. Gascuel. Spincas : a step toward virtual collaborative sculpting. *VRIC'02 Proceedings*, June 2002. (référencé page 134)

[GDG01] Raphael Grasset, Xavier Décoret, and Jean-Dominique Gascuel. Augmented reality collaborative environment :calibration and interactive scene editing. In *VRIC 2001*, May 2001. Laval Virtual. (référencé page 66)

[GG02] Raphael Grasset and Jean-Dominique Gascuel. Mare : Multiuser augmented reality environment on real table setup. In *ACM SIGGRAPH Conference Abstracts and Applications*, 2002. (référencé page 66)

[GH91] T.A. Galyean and J.F. Hughes. Sculpting : An interactive volumetric modeling technique. *Computer Graphics*, 25(4) :267–274, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991). (référencé pages 57, 58, 63, 77, 87, 91, 92, 95, 100, 101, 142, 143)

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *Proceedings of SIGGRAPH 97*, pages 209–216, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California. (référencé page 44)

[GH98] Michael Garland and Paul S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. *IEEE Visualization '98*, pages 263–270, October 1998. ISBN 0-8186-9176-X. (référencé pages 44, 130)

[GOP99] Carlos Gonzalez-Ochoa and Jörg Peters. Localized-hierarchy surface splines (less). *1999 ACM Symposium on Interactive 3D Graphics*, pages 7–16, April 1999. ISBN 1-58113-082-1. (référencé pages 17, 18, 19, 20, 138)

[HDD+94] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, pages 295–302, Orlando, Florida, July 1994. ACM SIGGRAPH / ACM Press. ISBN 0-89791-667-0. (référencé page 39)

[HH95] K. Hirota and M. Hirose. Providing force feedback in virtual environments. *IEEE Computer Graphics and Applications*, pages 22–30, September 1995. (référencé pages 74, 143)

[HHK92] W.M. Hsu, J.F. Hughes, and H. Kaufman. Direct manipulation of free-form deformations. *Computer Graphics (Proceedings of SIGGRAPH 92)*, 26(2) :177–184, July 1992. ISBN 0-201-51585-7. Held in Chicago, Illinois. (référencé pages 22, 23, 24, 86, 128, 138)

[HKD93] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using catmull-clark surfaces. *Proceedings of SIGGRAPH 93*, pages 35–44, August 1993. ISBN 0-201-58889-7. Held in Anaheim, California. (référencé pages 37, 139)

[Hop99] Hugues H. Hoppe. New quadric metric for simplifying meshes with appearance attributes. *IEEE Visualization '99*, pages 59–66, October 1999. ISBN 0-7803-5897-X. Held in San Francisco, California. (référencé page 44)

[HOYK99] M. Hirose, T. Ogi, H. Yano, and N. Kakehi. Development of wearable force display (hapticgear) for immersive projection displays. *Panel Session of the Proceedings of IEEE Virtual Reality*, March 1999. (référencé pages 73, 74, 143)

[IMT99] T. Igarashi, S. Matsuoka, and H. Tanaka. A sketching interface for 3d freeform design. *Computer Graphics*, pages 409–416, August 1999. Proceedings of SIGGRAPH'99 (Los Angeles). (référencé pages 75, 76, 77, 86, 100, 143)

[JLSW02] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *Proceedings of SIGGRAPH 2002*, page ? ?, July 2002. (référencé pages 131, 146)

[Kal93] R.S. Kalawsky. *The Science of Virtual Reality and Virtual Environments*. Addison-Wesley, 1993. (référencé pages 64, 65, 66, 73, 142, 143)

[KAW91] Z. Kačić-Alesić and B. Wyvill. Controlled blending of procedural implicit surfaces. *Graphic Interface'91*, pages 236–245, 1991. (référencé pages 50, 141)

[KB89] D. Kalra and A. H. Barr. Guaranteed ray intersections with implicit surfaces. *Computer Graphics*, 23(3) :297–306, July 1989. Proceedings of SIGGRAPH'89. (référencé page 51)

[KBSS01] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature-sensitive surface extraction from volume data. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 57–66. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1. (référencé pages 130, 131, 146)

[KCVS98] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. *Proceedings of SIGGRAPH 98*, pages 105–114, July 1998. ISBN 0-89791-999-8. (référencé page 14)

[KS99] Andrei Khodakovsky and Peter Schröder. Fine level feature editing for subdivision surfaces. *Proceedings of ACM Solid Modeling'99*, 1999. (référencé pages 46, 47, 140)

[KSW+99] Olaf Körne, Markus Schill, Clemens Wagner, H.J. Bender, and Reiner Männer. Haptic volume rendering with an intermediate local representation, 1999. First International Workshop on the Haptic Devices in Medical Applications. (référencé page 118)

[Lab97] Human Interface Technology Laboratory. Visual displays frequently asked questions (faq). `www.hitl.washington.edu/scivw/visual-faq.html`, Last modified 18/02/97. (référencé page 65)

[Lan97] E. Lantz. Future directions in visual display systems. *Computer Graphics*, pages 38–45, May 1997. (référencé pages 65, 81)

[Lat96] R. Latham. Force and tactile feedback in a virtual environment. *site* `www.cgsd.com/TOPIT.html`, November 1996. (référencé page 80)

[LC87] W.E. Lorensen and H.E. Cline. Marching cubes : a high resolution 3d surface construction algorithm. *Computer Graphics*, pages 163–169, July 1987. Proceedings of SIGGRAPH'87 (Anaheim). (référencé pages 52, 88)

[LDW97] Michael Lounsbery, Tony D. DeRose, and Joe Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1) :34–73, January 1997. ISSN 0730-0301. (référencé page 42)

[Lev99] Adi Levin. Interpolating nets of curves by smooth subdivision surfaces. *Computer Graphics*, pages 57–64, 1999. Proceedings of SIGGRAPH'99 (Los Angeles). (référencé pages 45, 46, 140)

[LPSD96] D. A. Lawrence, L. Y. Pao, M. A. Salada, and A. M. Dougherty. Quantitative experimental analysis of transparency and stability in haptic interfaces. *Proceedings of the 1996 ASME International Mechanical Engineering Congress and Exhibition, DSC-Vol. 58*, pages 441–449, November 1996. (référencé page 120)

[LSS+98] Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. Maps : Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH 98*, pages 95–104, July 1998. ISBN 0-89791-999-8. (référencé pages 42, 43, 140)

[Mah97] D.P. Mahoney. Visualizing volumes. *Computer Graphics World*, pages 42–48, January 1997. (référencé page 51)

[MBGN98] T. McReynolds, D. Blythe, B. Grantham, and S. Nelson. Advanced graphics programming techniques using opengl. *Computer Graphics*, July 1998. SIGGRAPH'98 Course Notes #17. (référencé pages 95, 145)

[MI87] S. Murakami and H. Ichihara. On a 3d display method by metaball technique. *Journal of papers at Electronics Communication Conference'87*, J70-D(8) :1607–1615, 1987. in Japanese. (référencé pages 49, 141)

[MJ96] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 181–188, New Orleans, Louisiana, August 1996. ACM SIGGRAPH / Addison Wesley. ISBN 0-201-94800-1. (référencé page 22)

[MK93] L. Moccozet and P. Kalra. Interactive and controlled synthesis of 3d irregular shapes. *Virtual World and Multimedia*, pages 179–198, 1993. also available at `miralabwww.unige.ch`. (référencé pages 11, 70)

[MOiT98] Shinji Mizuno, Minoru Okada, and Jun ichiro Toriwaki. Virtual sculpting and virtual woodcut printing. *The Visual Computer*, 14(2) :39–51, 1998. ISSN 0178-2789. (référencé page 56)

[MOyS+90] Margaret Minsky, Ming Ouh-young, Oliver Steele, Jr. Frederick P. Brooks, and Max Behensky. Feeling and seeing : Issues in force display. *1990 Symposium on Interactive 3D Graphics*, 24(2) :235–243, March 1990. ISBN 0-89791-351-5. (référencé pages 70, 122)

[MQ00] Kevin T. McDonnell and Hong Qin. Dynamic sculpting and animation of free-form subdivision solids. In *Computer Animation 2000*, pages 126–133. IEEE CS Press, May 2000. ISBN 0-7695-0683-6. (référencé pages 61, 62, 142)

[MQ02] Kevin T. McDonnell and Hong Qin. Dynamic sculpting and animation of free-form subdivision solids. *The Visual Computer*, 18(2) :81–96, 2002. ISSN 0178-2789. (référencé pages 61, 62, 142)

[MQW01] Kevin T. McDonnell, Hong Qin, and Robert A. Wlodarczyk. Virtual clay : A real-time sculpting system with haptic toolkits. *2001 ACM Symposium on Interactive 3D Graphics*, pages 179–190, March 2001. ISBN 1-58113-292-1. (référencé pages 61, 101)

[MS92] H.-P. Moreton and C.-H. Séquin. Functional optimization for fair surface design. *Computer Graphics*, 26(2) :167–176, July 1992. Proceedings of SIGGRAPH'92 (Chicago). (référencé pages 31, 32, 139)

[MS94] Thomas H. Massie and J. Kenneth Salisbury. The phantom haptic interface : A device for probing virtual objects, 1994. Proceedings of ASME'94. (référencé page 118)

[MT97] L. Moccozet and N.Magnenat Thalmann. Dirichlet free-form deformations and their application to hand simulation. *Proceedings of Computer Animation'97*, 1997. (référencé pages 23, 24, 138)

[Nay90] Bruce Naylor. Sculpt : An interactive solid modeling tool. In *Graphics Interface '90*, pages 138–148. Canadian Information Processing Society, May 1990. (référencé pages 11, 12, 137)

[NB93] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, November 1993. (référencé page 52)

[NHK+85] T. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. *Journal of papers at Electronics Communication Conference'85*, J68-D(4) :718–725, 1985. in Japanese. (référencé pages 49, 141)

[OC97] A. Opalach and M.-P. Cani. Local deformation for animation of implicit surfaces. *Proceedings of SCCG'97*, Jun 1997. Published under the name Marie-Paule Cani-Gascuel. (référencé pages 55, 56, 92, 141, 144)

[OHCD00] A.M. Okamura, M.W. Hage, M.R. Cutkosky, and J.T. Dennerlein. Improving reality-based models for vibration feedback. *American Society of Mechanical Engineers, Dynamic Systems and Control Division*, 69(2) :1117–1124, 2000. (référencé pages 121, 125, 146)

[Par77] Richard E. Parent. A system for sculpting 3-d data. *Computer Graphics (Proceedings of SIGGRAPH 77)*, 11(2) :138–147, July 1977. (référencé pages 9, 11)

[PASS95] A. Pasko, V. Adshiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling : concepts, implementation and applications. *The Visual Computer*, pages 429–446, 1995. (référencé page 50)

[PB88] J.C. Platt and A.H. Barr. Constraint methods for flexible models. *Computer Graphics*, 22(4) :279–288, August 1988. Proceedings of SIGGRAPH'88 (Atlanta). (référencé pages 28, 29, 139)

[PF01a] Ronald N. Perry and Sarah F. Frisken. Kizamu : A system for sculpting digital characters. *Proceedings of SIGGRAPH 2001*, pages 47–56, August 2001. ISBN 1-58113-292-1. (référencé pages 63, 101, 102, 110)

[PF01b] Jean-Paul Papin Philippe Fuchs, Guillaume Moreau. *Le traité de la réalité virtuelle*. Presses de l'Ecole des Mines de Paris, 2001. (référencé page 65)

[PHK+99] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L.Seiler. The volumepro real-time ray-casting system. *Computer Graphics*, pages 251–260, August 1999. Proceedings of SIGGRAPH'99 (Los Angeles). (référencé pages 52, 95)

[PK96] H.P. Pfister and A. Kaufman. Cube-4 - a scalable architecture for real-time volume rendering. *Computer Graphics*, pages 47–55, October 1996. Proceedings of Visualization'96 (San Francisco). (référencé pages 52, 95)

[PPK00] Ken Perlin, Salvatore Paxia, and Joel S. Kollin. An autostereoscopic display. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 319–326. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000. ISBN 1-58113-208-5. (référencé page 69)

[PS96] T. Poston and L. Serra. Dextrous virtual work. *Computer Graphics*, 39(5) :37–45, May 1996. (référencé pages 82, 83, 144)

[PSS01] A. Pasko, V. Savchenko, and A. Sourin. Synthetic carving using implicit surface primitives. *Computer Aided Design*, 33(5) :379–388, 2001. (référencé pages 57, 141)

[RCG+97] D. Rossin, M.-P. Cani, J.-D. Gascuel, A. Opalach, and M. Desbrun. Plateforme d'expérimentation pour la modélisation par surfaces implicites. *Proceedings of Modeleurs Géométriques'97*, September 1997. Published under the name Marie-Paule Cani-Gascuel. (référencé pages 92, 144)

[RE99] Alon Raviv and Gershon Elber. Three-dimensional freeform sculpting via zero sets of scalar trivariate functions. *Proceedings of the fifth symposium on Solid modeling and applications*, pages 246–257, June 1999. (référencé pages 60, 62)

[RE00] Alon Raviv and Gershon Elber. Three-dimensional freeform sculpting via zero sets of scalar trivariate functions. *Computer-Aided Design*, 32(8-9) :513–526, August 2000. ISSN 0010-4485. (référencé pages 60, 101)

[Ric73] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2) :157–160, 1973. (référencé page 51)

[RL00] Szymon Rusinkiewicz and Marc Levoy. Qsplat : A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 343–352. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 2000. ISBN 1-58113-208-5. (référencé page 130)

[RRS96] A. Rösch, M. Ruhl, and D. Saupe. Interactive visualization of implicit surfaces with singularities. *Proceedings of Implicit Surfaces'96 (Eindhoven)*, pages 73–87, October 1996. (référencé pages 48, 140)

[Sch83] C. Schmandt. Spatial input/display correspondence in a stereoscopic computer graphic work station. *Computer Graphics*, pages 253–261, July 1983. Proceedings of SIGGRAPH'83. (référencé pages 83, 144)

[SG97] Z. Szalavàri and M. Gervautz. The personnal interaction pannel : a two-handed interface for augmented reality. *Eurographics'97 Proceedings*, 16(3) :C–335–C–346, 1997. (référencé pages 81, 82, 144)

[SH97] B.T. Stander and J.C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. *Computer Graphics*, pages 279–286, August 1997. Proceedings of SIGGRAPH'97 (Los Angeles). (référencé pages 53, 54, 141)

[Sou01] Alexei Sourin. Functionally based virtual computer art. *2001 ACM Symposium on Interactive 3D Graphics*, pages 77–84, March 2001. ISBN 1-58113-292-1. (référencé pages 57, 77, 100, 130)

[SP86] T.-W. Sederberg and S.-R. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4) :151–160, August 1986. Proceedings of SIGGRAPH'86 (Dallas). (référencé pages 20, 21, 22, 138)

[SS99] Steven Schkolne and Peter Schröder. Surface drawing. *Technical Report Caltech Computer Science CS-TR-99-03*, 1999. (référencé pages 79, 143)

[SS01] Peter Schröder Steven Schkolne, Michael Pruett. Surface drawing : Creating organic 3d shapes with the hand and tangible tools. *Proceedings of CHI 2001*, March 2001. (référencé page 79)

[SSKK98] H. Suzuki, Y. Sakurai, T. Kanai, and F. Kimura. Interactive mesh dragging with adaptive remeshing technique. *Pacific Graphics '98*, pages 188–197, October 1998. Held in Singapore. (référencé pages 13, 137)

[ST92] R. Szeliski and D. Tonnesen. Surace modeling with oriented particles systems. *Computer Graphics*, 26(2) :185–194, July 1992. Proceedings of SIGGRAPH'92 (Chicago). (référencé pages 32, 33, 34, 35, 87, 139)

[Sta98] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 395–404, Orlando, Florida, July 1998. ACM SIGGRAPH / Addison Wesley. ISBN 0-89791-999-8. (référencé pages 38, 139)

[Tau95] Gabriel Taubin. A signal processing approach to fair surface design. *Proceedings of SIGGRAPH 95*, pages 351–358, August 1995. ISBN 0-201-84776-0. (référencé pages 35, 37, 39, 45)

[TC90] J.-A. Thingvold and E. Cohen. Physical modeling withB-splines surfaces for interactive design and animation. *Computer Graphics*, 24(4) :129–137, August 1990. Proceedings of SIGGRAPH'90 (Dallas). (référencé page 86)

[TF88] D. Terzopoulos and K. Fleischer. Modeling inelastic deformations : Viscoelasticity, plasticity, fracture. *Computer Graphics*, 22(4) :269–278, August 1988. Proceedings of SIGGRAPH'88 (Atlanta). (référencé pages 27, 28, 139)

[TPBF87] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4) :205–214, July 1987. Proceedings of SIGGRAPH'87 (Anaheim). (référencé pages 26, 27, 30, 139)

[TPF89] D. Terzopoulos, J. Platt, and K. Fleischer. Heating and melting deformable models (from goop to glop). *Proceedings of Graphic Interface'89*, pages 219–226, 1989. (référencé pages 28, 29, 139)

[Vas97] T.I. Vassilev. Interactive sculpting with deformable nonuniform b-splines. *Computer Graphics Forum*, 16(4) :191–199, 1997. (référencé page 86)

[VL00] A. Verroust and F. Lazarus. Extracting skeletal curves from 3d scattered data. *The Visual Computer*, 16(1) :15–25, 2000. (référencé page 128)

[WE98] R. Westermann and T. Ertl. Efficiently using graphics hardware in volume rendering applications. *Computer Graphics*, July 1998. (référencé page 52)

[Wes98] Stephen H. Westin. Computer-aided industrial design. *Computer Graphics*, 32(1), February 1998. (référencé page 116)

[WGG99] Brian Wyvill, Eric Galin, and Andrew Guy. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2) :149–158, June 1999. (référencé pages 55, 141)

[WH94] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics*, pages 269–278, July 1994. Proceedings of SIGGRAPH'94 (Orlando). (référencé pages 52, 53, 141)

[WK95] Sidney W. Wang and Arie E. Kaufman. Volume sculpting. *1995 Symposium on Interactive 3D Graphics*, pages 151–156, April 1995. ISBN 0-89791-736-7. (référencé pages 59, 63, 77, 87, 87, 91, 101, 110, 142)

[WMW86] B. Wyvill, C. McPheeters, and G. Wyvill. Data structure for soft objects. *Visual Computer*, 4(2) :227–234, August 1986. (référencé pages 49, 91, 141)

[WvGW94] O. Wilson, A. van Gelder, and J. Wilhelms. Direct volume rendering via 3d textures. *Technical Report UCSC-CRL-94-19*, June 1994. University of California, Santa Cruz. (référencé page 52)

[WW89] B. Wyvill and G. Wyvill. Field functions for implicit surfaces. *The Visual Computer*, 5 :75–82, December 1989. (référencé page 48)

[WW92] W. Welch and A. Witkin. Variatonal surface modeling. *Computer Graphics*, 26(2) :157–166, July 1992. Proceedings of SIGGRAPH'92 (Chicago). (référencé pages 31, 32, 139)

[WW94] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. *Computer Graphics*, pages 247–256, July 1994. Proceedings of SIGGRAPH'94 (Orlando). (référencé pages 33, 35, 36, 37, 87, 139)

[Zac00] Gabriel Zachmann. *Virtual Reality in Assembly Simulation – Collision Detection, Simulation Algorithms, and Interaction Techniques*. PhD thesis, Darmstadt University of Technology, summer 2000. Fraunhofer IRB Verlag. (référencé page 71)

[ZHH96] R.C. Zeleznik, K.P. Herndon, and J.F. Hughes. Sketch : An interface for sketching 3d scenes. *Computer Graphics*, pages 163–170, August 1996. Proceedings of SIGGRAPH'96 (New Orleans). (référencé pages 74, 86)

[ZSD+00] D. Zorin, P. Schröer, T. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for modeling and animation. *Computer Graphics*, July 2000. SIGGRAPH'00 Course Notes #23. (référencé pages 38, 39, 46, 47, 139)

[ZSS97] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. *Proceedings of SIGGRAPH 97*, pages 259–268, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California. (référencé pages 17, 44, 45, 140)