Modèles déformables pour l'animation : modélisation, animation et contrôle

Marie-Paule Gascuel

MEMOIRE d'HABILITATION

Présenté par

Marie-Paule Gascuel

pour obtenir le grade de

L'Habilitation à Diriger des Recherches
de
l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Spécialité : Informatique

Modèles Déformables pour l'Animation : Modélisation, Animation, et Contrôle

Date de soutenance : 29 mars 1995

Composition du jury :

Pr. Roger MOHR	Président
Pr. Daniel THALMANN	Rapporteur
Pr. Gérard HEGRON	Rapporteur
Pr. Claude PUECH	Rapporteur
Pr. Demetri TERZOPOULOS	Examinateur
Pr. Philippe CINQUIN	Examinateur

Habilitation préparée au sein du laboratoire :
iMAGIS/IMAG
Projet commun entre le CNRS, l'INRIA, l'Institut National Polytechnique de Grenoble et l'Université Joseph Fourier

Remerciements

Je tiens à adresser mes plus vifs remerciements aux membres de mon jury : Gérard Hégron, Daniel Thalmann et Claude Puech qui ont accepté d'être mes rapporteurs, Démétri Terzopoulos et Philippe Cinquin que je remercie pour l'intérêt qu'ils portent à mes travaux, ainsi que Roger Mohr qui me fait l'honneur de présider ce jury.

Les travaux décrits dans ce mémoire ont débuté au Laboratoire d'Informatique de l'Ecole Normale Supérieure à Paris, et se sont achevés à Grenoble, au sein de l'Institut IMAG. Merci beaucoup à Claude Puech, directeur d'iMAGIS, pour la peine qu'il s'est donné lors du transfert de notre équipe, et pour les excellentes conditions de travail qu'il nous a offertes. Merci également pour la confiance qu'il m'a accordée au cours de ces quatre années, et pour sa disponibilité sans faille pour écouter et résoudre les problèmes de chacun.

Je remercie très chaleureusement les étudiants dont j'ai dirigé les recherches, et sans qui ce travail aurait été impossible. Un merci tout particulier à Alexis Lamouret, Mathieu Desbrun, Nicolas Tsingos et Jean-Paul Smets pour leur enthousiasme et pour leur tenacité dans les moments difficiles. Merci également à Eric Bittar, avec qui j'ai beaucoup apprécié de travailler.

Enfin, un grand merci à Jean-Dominique Gascuel, François Sillion, et Georges Drettakis pour leur aide constante et le soutien qu'ils m'ont apporté.

Merci à tous les membres de l'équipe iMAGIS pour leur bonne humeur quotidienne, et en particulier à François, Jean-Christophe, Kevin, Fred, Rachel, Stéphane, Nicolas et Rosine.

Table des matières

Première partie

Synthèse : Modèles déformables pour l'animation

Introduction

Les méthodes traditionnelles d'animation par ordinateur, issues des techniques d'interpolation entre "position clés" du dessin animé, s'accordent mal avec la complexité sans cesse croissante des scènes de synthèse. En effet, ces méthodes sont purement descriptives : l'animateur[1] spécifie une série de positions et de formes pour un objet paramétré, et l'ordinateur se charge de calculer les images intermédiaires. Cette spécification ne produit qu'un seul mouvement (seul peut varier le mode d'interpolation), dont tous les aspects doivent être réglés "à la main". En particulier, il faudra éviter les interpénétrations avec d'autres objets de la scène, spécifier chaque variation de vitesse et chaque déformations. A l'opposé, des méthodes plus récentes tentent d'automatiser la synthèse de mouvements et de déformations. Typiquement, l'utilisateur associe à chaque objet un modèle physique simple, et laisse la machine calculer son mouvement et ses déformations en fonction des conditions initiales et des interactions éventuellement détectées avec d'autres objets. Particulièrement intéressants pour leur interface réduite, ces modèles sont "générateurs", en ce sens qu'ils ne codent non plus un seul mais *toute une classe de mouvements*, que l'on explore par simple modification des paramètres. Leur usage semble donc particulièrement bien adapté à l'animation d'objets complexes, et ce n'est pas un hasard si ce type de modèle s'est particulièrement développé pour l'animation de structures articulées et pour la modélisation de matériaux déformables, élastiques ou plastiques.

Bien qu'apparaissant depuis près de huit ans parmi les principaux thèmes de recherche en images de synthèse, les modèles générateurs[2] ne font toujours pas figure d'outil standard dans les logiciels d'animation du commerce. Ceci est d'autant plus surprenant que certaines avancées récentes en matière de modélisation notamment (Non Uniform Rational B-Splines, Free Form Deformations) ont été intégrées presque immédiatement dans les logiciels industriels. Le passage du stade expérimental à une utilisation concrète semble donc particulièrement délicat pour les modèles générateurs. Essayons de donner quelques

1. Dans tout ce document, on désignera par "animateur" tout utilisateur d'un logiciel d'animation, quelle que soit son expertise.

2. Dans tout ce document, je préfère utiliser ce terme plutôt que l'expression consacrée "modèles physiques", qui peut prêter à confusion dans la mesure où tous les modèles qui vont être évoqués ne sont pas issus de théories de la physique, et que, même lorsque c'est le cas, leur utilisation en synthèse d'images se fait souvent en dehors du domaines de validité des paramètres.

éléments d'explication.

- La principale difficulté concerne le mode de définition d'une animation qui est offert. Un animateur est en effet habitué à spécifier un mouvement en définissant des positions ou des formes clés pour les objets ou personnages à animer, conformément à un scénario donné. Spécifier un modèle physique pour chaque objet, le paramètrer correctement, puis le contrôler à partir de forces et de couples à appliquer au cours du temps semble relever de la gageure, surtout lorsqu'il s'agit de mouvements complexes, destinés par exemple à donner l'illusion de la vie.

  Même lorsque l'animation ne fait intervenir que des objets inertes, régler les paramètres du modèle physique en fonction de l'effet désiré n'a rien d'évident. En effet, il est rare que l'utilisateur trouve les paramètres auxquels il aimerait avoir accès. Peu de modèles déformables permettent, par exemple, d'imposer que les déformations conservent le volume. De plus, les quantités physiques qui paramètrisent le modèle peuvent n'avoir aucune signification pour un non physicien (c'est le cas par exemple des constantes de Lamé de la théorie de l'élasticité).

- En second lieu, comme nous le verrons en détail dans la section 1.1, les modèles développés jusqu'à présent ont le plus souvent été conçus pour représenter un comportement bien spécifique (élasticité linéaire ou non, plasticité, fractures, articulations à un nombre donné de degrés de liberté). Ce n'est donc pas un unique modèle, mais toute une bibliothèque de modèles qu'il faudrait intégrer, avec le risque d'oublier un type d'objet utile. De plus, utiliser un catalogue de modèles disparates peut imposer l'utilisation d'algorithmes d'animation et de traitement des interactions distincts, selon le ou les types d'objets concernés. Ainsi, intégrer un nouveau modèle peut demander l'écriture d'interfaces avec tous les autres (pour la prise en compte des interactions par exemple), ce qui augmente la lourdeur du système.

- Enfin, les pratiques de la production d'un film d'animation consistent généralement à mettre au point séparément un certain nombre de plans assez courts, sur lesquels on ne travaille pas forcément par ordre chronologique. Le plus souvent, les positions initiales des objets pour un plan donné correspondent aux positions finales du plan précédent. L'utilisation de modèles générateurs imposerait alors un travail en ordre chronologique, dans la mesure où ces positions finales ne sont déterminées qu'à l'issue d'une phase de simulation. De plus, un léger ajustement des paramètres servant au calcul de la première séquence pourrait complètement modifier les événements qui font suite, conduisant au re-calcul de toute l'animation (des divergences même très faibles entre deux mouvements calculés par un modèle générateur s'amplifient généralement au cours du temps). Ceci n'est pas la moindre des difficultés.

On peut citer un exemple qui illustre bien la difficulté d'utilisation des modèles générateurs pour réaliser une séquence, même courte, à scénario : il s'agit du

film illustrant l'article de David Baraff à la conférence *Siggraph* 1990 sur le traitement des collisions entre solides de forme libre [Bar90]. Le but était de simuler une série de chutes d'un petit cheval de bois, tombant d'une étagère à une autre en y renversant un certain nombre d'objets, pour atterrir finalement sur le clavier d'un ordinateur posé sur le bureau, plus bas. La méthode de travail adoptée par David Baraff pour mener à bien cette animation est particulièrement "parlante" : il a modélisé sa scène au fur et à mesure de la simulation, plaçant chaque nouvel objet à renverser au "bon endroit" en fonction de la position atteinte par le petit cheval au cours de sa chute, puis reprenant les calculs en amont pour juger de l'effet obtenu. Inutile de dire qu'une telle approche ne pourrait pas être retenue lors de la production d'un film commercial, dont le scénario a toute chance d'être imposé au programme d'animation, et non le contraire.

L'un des buts à long terme des travaux présentés dans ce mémoire serait de rendre les modèles générateurs plus facilement intégrables et utilisables au sein d'un logiciel d'animation du commerce. Pour cela, il s'agit de définir un cadre suffisamment général et modulable pour qu'une large variété d'objets puissent être définis en combinant des éléments existants, sans demander de développement spécifique. Chaque élément proposé, choisi pour la contribution qu'il peut apporter à la conception et au contrôle d'objets déformables complexes, doit rester facile à contrôler pour un animateur, généralement non spécialiste en mathématiques appliquées ou en physique. Pour faire face aux contraintes de l'interactivité, voire du temps réel, le calcul du mouvement et des déformations doit être aussi efficace que possible. Enfin, de manière à pouvoir passer aisément de la phase d'animation des objets au calcul du rendu d'une séquence, les modèles choisis doivent fournir une représentation surfacique utilisable des objets et de leurs déformations.

Ce document décrit la mise au point de structures prototypes d'objets complexes, en détaillant les trois aspects fondamentaux que sont la modélisation, l'animation, et le contrôle.

Le plan de ce mémoire est le suivant :

- Le chapitre 1 fait la synthèse des différentes approches développées jusqu'à présent pour la simulation d'objets déformables en synthèse d'images, et montre l'intérêt d'une architecture hybride pour construire et contrôler facilement des objets déformables complexes.

- Le chapitre 2 présente la structure interne de base que nous utilisons pour la synthèse du mouvement. Elle permet de spécifier les propriétés dynamiques fondamentales d'un objet : masse, inertie, ainsi qu'articulations dans le cas d'une structure composite. L'objet est alors construit comme un réseau de repères locaux, chacun muni de sa masse et de son inertie, reliés entre eux par des contraintes géométriques. Eventuellement munies de butées, ces contraintes articulaires sont maintenues au cours de la simulation par une série de corrections en déplacement.

- La structure externe, qui offre une définition surfacique des objets et sert d'interface avec le monde extérieur, est présentée dans le chapitre 3. Il

s'agit d'une couche de matériau rigide ou déformable, dont le déplacement est contrôlé par la structure interne. L'originalité du modèle réside dans la formulation implicite que nous proposons, et qui offre une représentation compacte et unifiée de la géométrie de l'objet et de ses caractéristiques physiques (élasticité linéaire ou non-linéaire en particulier). Capable de détecter efficacement les collisions avec les autres objets de la scène, la structure externe se déforme en conséquence, et transmet les forces de réaction aux composants internes.

- Le chapitre 4 décrit la couche de contrôle qui peut être ajoutée aux deux modules précédents de manière à permettre la synthèse de séquences "à scénario". Il s'agit alors de combiner simulation – et en particulier détection et traitement automatiques des collisions – et contrôle partiel de trajectoire. L'utilisateur spécifie une ébauche de trajectoire sous forme de positions clés. Cette trajectoire est corrigée au cours du mouvement en fonction des propriétés dynamiques des objets et des événements qui sont détectés. Le modèle générateur devient ainsi un véritable outil d'aide au réalisme, utilisable conjointement à une interface traditionnelle.

- Le chapitre 5 regroupe différentes applications du formalisme général qui a été mis en place. La principale d'entre elles est la construction de modèles pour l'animation de synthèse. D'autres travaux débouchent sur la reconstruction semi-automatique d'organes à partir de données médicales. A terme, ces modèles devraient pouvoir être utilisés dans le cadre de simulations. Enfin, des collaborations avec d'autres instituts de recherche débutent actuellement: il s'agit d'une part d'appliquer les modèles développés à la simulation de lèvres parlantes, et d'autre part d'une étude de trajectographie de blocs sur des terrains déformables.

Nous terminons cette synthèse par un bilan des perspectives offertes par les modèles générateurs dans le cadre de l'animation de synthèse et de ses applications.

Chapitre 1

Conception de modèles générateurs

1.1 Modélisation d'objets déformables

Avertissement

Le but de ce qui suit n'est pas de constituer un nouvel état de l'art sur l'animation d'objets déformables. En effet, deux études particulièrement complètes sur les modèles générateurs en animation de synthèse ont été publiées récemment en France, dans le cadre de la thèse de Stéphane Jimenez [Jim93] d'une part, et dans le mémoire d'habilitation de Bruno Arnaldi [Arn94] d'autre part. Il reste cependant important de rappeler brièvement quelles sont les principales approches en animation d'objets déformables, de manière à éclairer le cheminement qui m'a conduit a conception des modèles présentés dans ce mémoire.

Une classification des modèles déformables en animation

Les modèles générateurs, généralement basés sur certaines lois de la physique, permettent de calculer l'évolution d'un objet au cours du temps, ce qui est fait le plus souvent en intégrant numériquement une équation différentielle associée au modèle. En ce qui concerne les objets déformables, on désigne par "élastique" un objet qui reprend sa forme initiale après toute déformation, par opposition aux objets "inélastiques" dont la forme courante dépend de l'historique des déformations qu'ils ont subies. Les matières plastiques, ainsi que les matériaux pouvant subir des fractures sont des cas particuliers d'objets inélastiques.

Les modèles déformables en animation de synthèse peuvent être regroupés en plusieurs catégories, dont les caractéristiques sont détaillées dans les paragraphes qui suivent. Nous utilisons la classification suivante :

1. Approches nodales :
   - équations continues discrétisées,
   - modèles mécaniques discrets.

2. Approches globales.

Une description détaillée de la plupart des modèles qui sont évoqués ici est donnée dans [GP91], inclus page 55 de ce mémoire. Cependant, les approches dites "globales" n'y figurent pas. Ceci illustre l'évolution extrêmement rapide du domaine.

1. Approches nodales

La plupart des modèles déformables utilisés en synthèse d'images expriment les déformations d'un objet par l'intermédiaire du déplacement de "nœuds" à l'intérieur de ce dernier. Ces nœuds sont soit les sommets d'un maillage utilisé pour la formulation discrète d'équations continues, soit des masses ponctuelles ayant explicitement servi à la création du modèle. On peut ainsi distinguer deux types d'approches nodales :

Equations continues discrétisées : Il s'agit de partir d'une équation donnée par la physique pour la modélisation d'un comportement précis, par exemple l'élasticité linéaire ou non linéaire. De cette équation éventuellement simplifiée et parfois même utilisée hors de son domaine de validité, est dérivé un mode de résolution discrète (double discrétisation spatiale[1] et temporelle), d'ou l'on tire le calcul du mouvement et des déformations de l'objet. Dans la pratique, un système matriciel représentant le comportement de l'objet discrétisé doit être résolu à chaque pas de temps [TPBF87, GMTT89, Dum90].

Un certain nombre d'auteurs se sont par la suite éloignés de l'utilisation directe des équations de la physique, pour construire des modèles plus spécifiques à l'animation de synthèse [TW88, TF88, TPF89]. Cela leur a permis de généraliser la simulation à des comportements plus variés, comme en particulier la plasticité et les fractures. Bien que les résultats visuels obtenus restent très convaincants, on peut regréter le fait que ces ces adaptations conduisent à un modèle, voire à un algorithme de simulation particulier pour chaque comportement.

Modèles mécaniques discrets : D'autres approches nodales procèdent au contraire de bas en haut. Au lieu de partir d'une équation toute faite, forcément spécifique au comportement qu'elle modélise, il est possible de construire progressivement le modèle d'un objet en reliant entre eux différents composants mécaniques de base. Les travaux d'Annie Luciani [Luc85, LJF+91] développent ce formalisme : tout objet sera représenté par un ensemble de masses élémentaires reliées par diverses fonctions d'interaction. Le calcul du mouvement et des déformations se fait alors *indépendamment sur chaque masse*, par résolution des équations de la mécanique du point. Ce type de méthode, qui favorise la mise en parallèle

1. La discrétisation spatiale se fait généralement par des techniques de différences finies, ou parfois d'éléments finis.

des calculs, est donc bien différente algorithmiquement de la classe précédente, pour laquelle les positions de tous les nœuds étaient déterminées simultanément.

Pour un premier type d'applications (objets visco-élastiques, ou visco-plastiques sans changement de forme important), les masses ponctuelles sont souvent agencées en réseau d'inter-connexions de topologie fixe, ce qui permet de effectuer rapidement le bilan des forces sur chaque masse [Mil88, DAH89, CHP89, Jim93]. Les liaisons utilisées sont le plus souvent des ressorts ou des éléments plastiques, montés en série ou en parallèle avec des liaisons de type frottement visqueux. Notons qu'un formalisme similaire, également à base de réseaux de masses ponctuelles, a été utilisé par Cornélius van Overveld [Ove90, Ove91] pour obtenir une simulation approchée de systèmes rigides articulés.

Pour modéliser des objets inélastiques pouvant subir des déformations importantes et des changements de topologie comme des fractures ou des fusions, on utilise ce que l'on appelle des *systèmes de particules.* Il s'agit d'ensembles de masses ponctuelles sans topologie fixe. Chaque masse interagit avec toutes les autres, selon des lois fonction de la distance. On trouve le plus souvent des modèles d'interaction dérivés des lois inter-particulaires "de Lennard-Jones", qui comprennent une attraction à moyenne portée et une répulsion à courte portée [MP89, LJR+91, Ton91, Jim93]. Le bilan des forces sur chaque masse, à priori coûteux ($O(n^2)$ où n est le nombre de particules), peut être ramené à $O(n\log(n))$, ou même à $O(n)$ lorsque les rayons d'action des forces sont limités, grâce à un tri spatial des données (voir [Ton91]).

Remarquons que le formalisme qui consiste à modéliser un objet à partir d'éléments mécaniques discrets (masses et lois d'interaction) est en définitive plus général que l'approche descendante du paragraphe précédent. En effet, l'algorithme d'animation est ici toujours le même, et différents comportement sont obtenus par simple modification des lois d'interaction entre masses. Cette conjonction d'éléments simples peut faire apparaître des comportements "émergeants" extrêmement complexes. Par exemple, un ensemble de particules soumis à de fortes contraintes se fracturera éventuellement en plusieurs blocs, sans qu'il soit nécessaire de mettre au point une modélisation spécifique des lieux de fracture comme dans [TF88].

2. Approches globales

Cette seconde grande classe de méthodes est apparue plus tardivement. Elle regroupe des techniques permettant d'optimiser le calcul des déformations en les restreignant à des classes particulières de transformations. Ainsi, au lieu de modéliser la propagation progressive d'une déformation à travers les nœuds de discrétisation du modèle, la géométrie de ce dernier est déformée en une seule étape, par exemple par application d'une matrice de déformation (comme celles définies dans [Bar84]). Ce raccourci permet la suppression des modes de déformation de haute fréquence (vibrations rapides mais de faible amplitude), ce qui

permet de relever l'intervalle de temps de simulation et d'obtenir l'interactivité, voire le temps réel, même pour des scènes complexes.

Alex Pentland [PW89] et Andrew Witkin [WW90] présentent deux modèles différents qui s'inscrivent dans ce cadre. Le premier part de l'expression des modes vibratoires d'un objet déformable, tandis que le second restreint l'espace des déformations à un certain type de transformations matricielles. Ces deux modèles produisent uniquement des déformations globales (compression, courbure, torsion de tout l'objet). L'approche développée par Dimitri Metaxas et Demetri Terzopoulos [MT91] permet de combiner ce type de modèle avec une structure maillée permettant la superposition de déformations locales.

Notons que du fait même de leur formulation, les approches globales sont adaptées aux objets visco-élastique homogènes, et ne semblent pas permettre la modélisation de changements de topologie.

Collisions et contacts entre objets déformables

Les interactions entre objets jouent un grand rôle dans l'animation par modèles générateurs, puisqu'elles sont la source de la plupart des mouvements et déformations que l'on peut observer dans un film de synthèse. Parce que très peu étudiées dans les ouvrages de physique (qui s'intéressent surtout aux états d'équilibre et aux petites oscillations [Cia85]), les collisions entre objets déformables ont souvent fait l'objet de solutions spécifiques à la synthèse d'images. La difficulté qu'il y a à proposer un algorithme "raisonnable" et assez général est augmentée par le fait que ces collisions, contrairement aux chocs entre objets rigides, ne sont en général ni conservatives ni instantanées. Du fait de leur durée dans le temps, il paraît essentiel de choisir des modèles offrant une modélisation fine des déformations dans les zones de contact, de manière à ce que tout au moins, l'animation obtenue soit visuellement crédible.

Avant toute chose, la notion même de "contact" entre objets déformables pose problème. Comme nous l'avons dit plus haut, un objet déformable n'est souvent qu'un réseau de nœuds, ou un ensemble discret de masses ponctuelles, pour lesquelles la notion de "surface" n'apparaît pas de manière naturelle. Si on s'intéresse à l'aspect visuel, un "contact" peut être défini au niveau des représentations géométriques des objets qui sont visualisées à chaque instant :

- Pour un modèle d'objet qui ne change pas de topologie, on affiche généralement soit une triangulation reposant sur les nœuds "extérieurs" du maillage de l'objet [TPBF87, GMTT89], soit une surface spline construite à partir de ces nœuds [Mil88, GVP91].

- Les systèmes à particules, sans topologie fixe, n'offrent aucune notion de "nœud extérieur" puisque toute masse élémentaire peut passer de la périphérie à l'intérieur de l'objet, et vice versa. Une solution pour la définition d'une surface pourrait consister à trianguler l'objet à chaque étape de l'animation (en construisant par exemple une triangulation de Delaunay). Cependant, cette méthode ne semble pas avoir été utilisée, probablement du fait de son coût et peut-être des fluctuations qu'elle pourrait engendrer

au niveau de la surface. En conséquence, certains auteurs visualisent simplement un ensemble de sphères donnant une idée des zones de répulsion autour de chaque particule [LJF+91, LJR+91], ce qui permet de mettre l'accent sur l'esthétique pure du mouvement. D'autres affichent une surface implicite isopotentielle[2] dépendant continuement des positions des particules [TPF89, Ton91].

Assez curieusement, la définition d'un "contact" utilisée dans les modules de détection des collisions n'est pas toujours cohérente avec ce choix d'une représentation géométrique pour l'objet.

- Lorsque les surfaces affichées sont des splines [Mil88, GVP91], l'utilisation de l'équation paramétrique exacte des surfaces apparaît comme trop coûteuse. Les auteurs se rabattent alors sur une détection basée sur le polygone de contrôle définissant ces surfaces, solution somme toute assez imprécise (en effet, une collision peut être détectée aussi bien avant le contact visuel qu'après l'interpénétration, selon le type de spline utilisé).

- Une solution plus satisfaisante peut être mise en œuvre lorsque les deux objets sont facétisés [MW88]. Si leurs boîtes englobantes s'intersectent, la zone du choc est déterminée en regardant si la trajectoire d'un sommet pendant un intervalle de temps n'a pas intersecté la trajectoire de chaque facette de l'autre objet pendant ce même intervalle. Cela conduit à la résolution d'équations de degré 5 pour les $O(n^2)$ couples sommet–facette, où n est le nombre total de points de discrétisation des deux surfaces.

- Comme le montre Alex Pentland [PW89], les surfaces implicites sont particulièrement bien adaptées à une détection précise et rapide des collisions. En effet, ce mode de représentation est basé sur une fonction "dedans–dehors" qui permet de dire en une seule étape de calcul si un point donné de l'espace est à l'intérieur ou à l'extérieur d'un solide (voir chapitre 3). Toujours après une phase de pré-détection utilisant des boîtes englobantes, l'un des objets est discrétisé et l'on teste si les points obtenus sont dans l'autre objet ou non. Ce procédé donne un coût en $O(n)$ seulement[3].

Détecter précisément les collisions à partir de la surface d'un objet n'est qu'un premier pas. Il reste encore à y répondre, en calculant les forces et les déformations qui permettent d'éviter l'interpénétration et de produire le mouvement après le choc. La plupart des méthodes utilisées jusqu'à présent dans le cadre des collisions entre objets déformables sont détaillées dans [GP91] (inclus page 55 de ce mémoire). Deux d'entre elles sont particulièrement intéressantes, à des titres divers :

- La méthode de pénalisation présentée dans [MW88] est probablement la plus la plus utilisée. On la retrouve sous des formes variées dans [TPBF87,

2. Ce type de surface sera défini précisément au chapitre 3.

3. Notons que les surfaces implicites ont été parfois utilisées uniquement pour de la phase de rendu final, sans qu'aucun profit ne soit tiré de leurs fonctions dedans-dehors lors des étapes de calcul [TPF89, Ton91].

TW88, TF88, PW89, DAH89]. L'idée de base consiste à utiliser la profondeur de l'interpénétration entre deux objets pour calculer la force de réaction à appliquer à l'intervalle de temps suivant. Ce calcul, qui a l'avantage d'être simple et universel, va malheureusement de pair avec des déformations incorrectes, puisque une interpénétration locale aura lieu pendant toute la durée du contact[4].

- La méthode analytique de [BW92] fait suite aux travaux de Dave Baraff sur les interactions entre solides rigides [Bar89, Bar90, Bar91]. Adaptée au modèle déformable d'Andrew Witkin et de William Welsh [WW90], cette technique combine l'emploi d'impulsions pour empêcher les interpénétrations à l'instant d'un choc, et une formulation sous forme de forces de réaction lors des phases de contact prolongé. Cette méthode, qui a l'avantage d'engendrer des déformations correctes des objets, est malheureusement limitée : elle ne s'applique que dans les situations où l'interpénétration peut être évitée en appliquant des forces en un nombre fini de points (objets polyhédriques). Par exemple, un cylindre posé sur un cube, ou un contact entre objets déformables de forme lisse, ne peuvent être modélisés de cette façon.

Remarquons enfin le rôle particulier joué par les systèmes particulaires : s'ils ne conduisent pas à une définition surfacique des objets, ils proposent par contre un modèle très cohérent pour la réponse aux collisions. En effet, ces dernières seront prises en compte tout naturellement au cours d'une simulation, en tant qu'interactions entre masses provenant de deux objets distincts.

Bilan

S'ils ont souvent produit des résultats très impressionnants, les modèles issus de la physique des milieux continus ont le défaut de rester très spécifiques à une application particulière. Dans le cadre d'un système d'animation de synthèse devant intégrer une multitude d'objets aux comportement variés, les modèles mécaniques discrets, construits par agencements successifs d'éléments simples, semblent être plus prometteurs. Malheureusement, ces modèles ne conduisent pas de manière naturelle à la production d'images, dans la mesure où il n'offrent pas directement de *représentation surfacique* des objets. Notons que pour permettre une détection et un traitement corrects des contacts, une telle représentation devrait être utilisée tout au long du processus d'animation.

1.2 Mise en place d'une architecture modulaire

Buts à atteindre

L'objectif de mes travaux est de proposer un cadre général permettant la définition de modèles pour l'animation d'objets complexes, et en particulier

4. Parfois, on entoure au contraire chaque obstacle d'une zone de répulsion, mais le rebond d'un objet peut alors avoir lieu avant même qu'il y ait contact visuel.

d'objets déformables.

- Pour décharger l'animateur de la spécification du mouvement et des déformations dans le cas d'objets complexes (déformables, articulés, souvent non-homogènes), chaque modèle devra être *générateur*. Notons que cela n'impose pas à priori d'opter pour un "modèle physique" au sens strict du terme.

- En vue de son utilisation en synthèse d'images, un modèle doit permettre de passer rapidement de la phase de modélisation d'un objet à son animation, puis à son rendu final. En particulier, la représentation des objets déformés sur laquelle se base le calcul de l'éclairage doit être si possible la même que celle utilisée pendant le calcul de l'animation, de manière à limiter les pertes de précision sur le rendu des surfaces en contact. Cela conduit à représenter la géométrie des objets sous forme de *surfaces déformables complexes* tout au long de la synthèse du mouvement et des déformations.

Ces deux choix fondamentaux étant établis, il reste à préciser les principales qualités que l'on aimerait offrir pour la modélisation d'objets complexes :

Généricité : le cadre de définition d'un objet doit être suffisamment général pour pouvoir s'appliquer à la plupart des comportements que l'on désire simuler en synthèse d'images. Le passage d'un type d'objet à un autre doit se faire par simple choix des modules à combiner, de leurs paramètres, et des éléments de liaison, sans qu'aucune programmation ne soit nécessaire en général.

Compacité : les paramètres doivent être simples et aussi peu nombreux que possible pour faciliter "l'apprentissage" du modèle par un utilisateur, qui ne sera à priori spécialiste ni des mathématiques appliquées, ni de la physique. Réduire le nombre de paramètres permet également de stocker les objets et leurs déformations de manière plus compacte.

Efficacité : une animation interactive facilite la mise au point progressive d'un modèle d'objet. Le critère d'efficacité peut être fondamental pour certaines applications (imagerie médicale, réalité virtuelle).

Notons que la recherche de structures d'objet compactes et permettant une simulation efficace conduit à s'éloigner du domaine de la physique des milieux continus. Nous sommes donc amenés à mettre au point une stratégie de conception bien *spécifique à la synthèse d'images.*

Choix d'une approche

En ce qui concerne le choix d'un formalisme général, j'ai été particulièrement marquée par les idées de fond défendues dans deux papiers :

- Dans [LC86], Annie Luciani et Claude Cadoz affirment que l'important n'est pas définir le véritable "modèle physique" des objets à simuler, mais

d'en trouver une *représentation mécanique* la plus simple possible, et qui soit bien adaptée à l'application mise en œuvre. Ainsi, ils proposent de construire chaque objet sous la forme d'un réseau à trois couches : masses ponctuelles, éléments transmetteurs, et points géométriques. La simulation est ainsi rendue plus économique, puisqu'elle ne porte que sur la première couche. Des calculs purement géométriques permettent d'en déduire la position des points à afficher, via les éléments transmetteurs.

- Dans [CHP89], John Chadwick insiste sur la facilité de contrôle qu'offre une *structure modulaire* d'objet, qui permet de gérer les paramètres de différentes couches de manière totalement indépendante. Ainsi, ils proposent de structurer un modèle de personnage en : un système de contrôle du mouvement ; un squelette rigide articulé ; des boîtes de déformation FFD[5] dont la forme est dictée soit par les mouvements articulaires du squelette, soit par un système masses-ressorts ; une "peau", surface géométrique contrôlée par les FFDs. Sur chacune de ces couches, l'utilisateur peut choisir son niveau d'intervention, d'un haut niveau de contrôle à une modification "manuelle" des paramètres, parfois utile pour dicter très précisément une déformation particulière.

De nombreux autres papiers adoptent également une structure hybride d'objet dans le cadre de l'animation par modèles générateurs [TF88, TW88, Mil88, TPF89, For91, GVP91] (la plupart de ces modèles sont décrits dans [GP91], inclus à la page 55 de ce mémoire). Bien sur, une architecture modulaire n'implique pas qu'un modèle soit général. Cependant, ces travaux semblent indiquer qu'en définitive, mieux vaut combiner entre elles plusieurs couches simples astucieusement choisies et contrôlées indépendamment les unes des autres que de programmer un modèle spécifique pour chaque application.

En conséquence, le travail décrit dans ce mémoire opte pour la mise en place d'une *structure modulaire* (ou *hybride*) pour les objets déformables. Un certain nombre de composants de base, qu'il s'agit de définir de manière suffisamment générale, pourront être agencés entre eux de manière à réaliser des structures complexes, sans qu'il soit nécessaire de revoir l'algorithme d'animation. Le contrôle sera facilité par l'accès indépendant aux paramètres des différentes couches, qui pourront être testées isolément avant l'assemblage final[6].

Structure proposée pour la définition d'objets complexes

Si une réserve peut être formulée sur les deux structures modulaires citées page 17 [LC86, CHP89], elle concerne la structuration des couches mises en

5. Free Form Deformations [SP86]. Cette technique consiste à placer un modèle géométrique dans une boîte déformable maillée, dont la forme est gérée par le déplacement d'un réseau de points de contrôle.

6. Dans la pratique, cette méthode de travail est mise en œuvre par tous les membres de l'équipe iMAGIS qui travaillent sur l'animation. Chaque module est défini dans le cadre de la plateforme d'animation orientée objet *Fabule*, développée en C++ sous la responsabilité de Jean-Dominique Gascuel [Gas94]. Les programmes de tests sont écrits dans le langage interprété Tcl, de manière à faciliter l'expérimentation.

place. En effet, l'architecture utilisée dans les deux cas est purement hiérarchique. Les paramètres des différentes couches se déduisent les uns des autres, du niveau le plus "interne" vers la couche "externe" (éléments à la surface de l'objet), sans qu'aucune possibilité de *rebouclage* ne soit offerte. Si nous prenons l'exemple du modèle de personnage de Chadwick [CHP89], y inclure un module de détection et de traitement automatiques des collisions serait particulièrement difficile, puisqu'une collision détectée au niveau de la surface devrait logiquement affecter le mouvement et les déformations ultérieures du squelette, ce qui est impossible du fait de la méthode adoptée.

Nous proposons donc un emploi différent d'une architecture modulaire : chacune des couches de notre modèle, qu'elle soit génératrice de mouvements ou de déformations, sera capable de communiquer avec le reste de la structure. En particulier, le modèle de surface déformable utilisé pour représenter la géométrie des objets, qui est de toute évidence le plus apte à détecter les collisions et à calculer les déformations ainsi que les forces de réaction qui en résultent, aura la possibilité de communiquer ces forces aux éléments internes, pour qu'elles soient prises en compte dans le calcul du mouvement au pas de temps suivant.

Par exemple, un objet élastique inerte, capable de déformations locales, pourra être représenté par deux modules s'échangeant des informations au cours de l'animation (cette structure a été utilisée dans [Gas93], inclus page 93 de ce mémoire) :

- un composant interne, utilisé pour calculer le mouvement à partir des propriétés dynamiques de l'objet (masse, inertie), et transmettant les informations de position et d'orientation à la structure externe ;

- un composant externe, représentant la matière élastique qui constitue l'objet, qui détecte les interactions avec l'extérieur, calcule les déformation locales qui en résultent, et transmet les forces de réaction à la structure interne.

Notons que cette décomposition en deux modules interne/externe, où le mouvement est calculé au niveau interne et les déformations au niveau externe, a été utilisée par plusieurs modèles précédents [TW88, TF88, GVP91]. Il s'agit bien d'une approximation, puisque les variations de la distribution de masse dues aux déformations locales ne sont pas prises en compte lors du calcul du mouvement.

Des objets plus complexes peuvent être construits en diversifiant la composition des couches utilisées, ainsi que leur nombre. Par exemple, l'utilisation d'une structure interne rigide articulée permet l'animation de modèles simplifiés de personnages ; une structure interne particulaire favorise par contre la synthèse de substances inélastiques, à grands champs de déformation, et capables de se fracturer au cours de l'animation. Enfin, d'autres couches peuvent être ajoutées au modèle, en particulier des modules de contrôle (qui peuvent résoudre des problèmes aussi différents que la gestion des variations de volume au cours d'une déformation, ou le contrôle de la trajectoire d'un ou plusieurs objets).

La suite de ce document de synthèse passe en revue les éléments qui viennent d'être évoqués : les différentes structures internes pour la synthèse des mouvements, la structure externe ainsi que les problèmes de détection et de traitement des interactions qui lui sont associées, ainsi que l'intégration de modules de contrôle. Le chapitre portant sur les application donne des exemples concrets d'agencement de ces différents composants.

Chapitre 2

Composants internes : simulation dynamique sous contraintes

Comme nous l'avons indiqué au chapitre précédent, la "structure interne" (ou les "composants internes") d'un objet regroupe tous les paramètres qui sont nécessaires pour calculer son mouvement à partir de la donnée de son état courant (position et vitesse) ainsi que des forces extérieures qui lui sont appliquées. En particulier la masse et le tenseur d'inertie de chaque composant de l'objet, ainsi que la description des connexions éventuelles qui les relient sont inclues dans cette notion.

Ce chapitre présente tout d'abord les structures de base que sont les points matériels et les solides, puis étudie différentes manières de les relier entre eux de manière à pouvoir créer des réseaux complexes. Les connexions utilisées peuvent être définies soit sous forme de forces d'interaction (on peut ainsi construire un système de particules, ou un réseau masses/ressorts de topologie fixe), soit sous forme de contraintes géométriques. Nous présentons pour cela un algorithme original d'animation sous contraintes, qui permet de construire et animer des structures rigides articulées dont chaque liaison a un nombre quelconque de degrés de liberté, éventuellement munis de butées. Le graphe des objets contraints peut comporter des boucles fermées.

2.1 Simulation de la dynamique

Ce paragraphe rappelle les équations fondamentales de la dynamique du point et du solide, et indique les schémas d'intégration que nous avons utilisés, ce qui permet aussi de fixer les notations que l'on retrouvera tout au long de ce mémoire.

Equations fondamentales du mouvement

Physique du point : Un point matériel est défini par sa position x, et sa masse m. Son animation s'effectue un intégrant numériquement au cours

du temps l'équation :

$$F = m\ddot{x}$$

où F représente la somme des forces extérieures appliquées au point.

Physique du solide : Un solide est défini par la donnée d'un repère local (matrice d'orientation R et position x par rapport au repère du monde), ainsi que par sa masse m et sa matrice d'inertie J. Son mouvement est régi par le système d'équations :

$$\begin{aligned} F &= m\ddot{x} \\ \mathcal{M} &= J\dot{\omega} + \omega \wedge J\omega \end{aligned}$$

où $\mathcal{M}$ est la somme des couples appliqués au solide, et ω son vecteur vitesse angulaire.

Schémas d'intégration

Deux schémas d'intégration des équations du mouvement ont été successivement testés au cours de ces recherches.

Schéma d'Euler : Il s'agit d'une intégration du premier ordre des équations du mouvement :

$$\begin{aligned} \dot{x}(t+dt) &= \dot{x}(t) + \frac{F(t)}{m}\, dt \\ x(t+dt) &= x(t) + \dot{x}(t+dt)\, dt \\ \omega(t+dt) &= \omega(t) + J^{-1}\left(\mathcal{M}(t) - \omega(t) \wedge J\omega(t)\right)\, dt \\ R(t+dt) &= \left(I + dt\, \tilde{\omega}(t+dt)\right) R(t) \end{aligned}$$

où $\tilde{\omega}$ est la matrice vitesse de rotation associée à ω. Cette matrice est caractérisée par : $\forall a \in \mathbb{R}^3 \;\; \tilde{\omega}a = \omega \wedge a$.

Schémas de Newton : Il est conçu de manière à donner une solution exacte pour des forces et couples qui seraient constants pendant un intervalle de temps dt (voir [PTVF92]). Nous avons opté pour ce schéma depuis 1993.

$$\begin{aligned} \dot{x}(t+dt) &= \dot{x}(t) + \frac{F(t)}{m}\, dt \\ x(t+dt) &= x(t) + \frac{1}{2}(\dot{x}(t) + \dot{x}(t+dt))dt \\ \omega(t+dt) &= \omega(t) + J^{-1}\left(\mathcal{M}(t) - \omega(t) \wedge J\omega(t)\right)\, dt \\ R(t+dt) &= \left(I + \frac{1}{2}(\tilde{\omega}(t) + \tilde{\omega}(t+dt))\, dt\right) R(t) \end{aligned}$$

2.2 Réseaux d'interactions

Dans la pratique, il est intéressant de combiner entre elles les structures génératrices de base, points matériels ou solides, de manière à créer des architectures d'objets complexes. Une première méthode pour ce faire consiste à

relier les différents composants par des fonctions d'interaction. Dans ce cadre, deux types de réseaux peuvent être crées :

- des réseaux de topologie fixe, pour lesquels chaque composant est relié à un ensemble fixé (et à priori restreint) de voisins :
- des systèmes de particules, pour lesquels chaque composant peut interagir avec tous les autres (dans la pratique, on choisit souvent des lois d'interaction dont le rayon d'action est limité, de manière à éviter les calculs sur un graphe d'interactions complet).

Dans les deux cas, l'intégration des équations du mouvement se fait individuellement, élément par élément, en utilisant les schémas donnés au paragraphe précédent. Simplement, les forces extérieures engendrées par les connexions sont prises en compte dans la somme des forces.

Tandis que les connexions en réseau fixe sont les mieux adaptés à la modélisation d'objets structurés, dont la forme ne varie par trop (le stockage d'une liste pré-calculée de voisins pour chaque masse est dans ce cas le plus efficace), les systèmes de particules permettent la modélisation de corps déformables à topologie variable. Nous avons eu l'occasion d'expérimenter ces systèmes dans le cadre de la synthèse de substances à grands champs de déformation (voir [DG94], inclus page 102 de ce mémoire). Les forces d'interaction que nous utilisons pour cette application comprennent des forces inter-particulaires dites "de Lennard-Jones" (composées d'une attraction à moyenne portée et d'une répulsion à faible portée), ainsi qu'un frottement visqueux que nous avons introduit, fonction de la distance entre deux particules et du carré de leurs vitesses relatives. Ce frottement, proportionnel pour chaque particule à la densité locale du système autour d'elle, nous permet de prendre en compte la déperdition d'énergie au sein d'un matériau visqueux.

2.3 Réseaux de contraintes géométriques

Si les réseaux d'inter-connexions permettent l'animation d'objets étirables, voire malléables, ils conviennent mal à la synthèse de "structures articulées", c'est à dire d'objets constitués de composants rigides reliés par des articulations. Utiles en animation de personnages, les structures articulées ont fait l'objet de nombreux travaux [AG85, IC87, IC88, PB88, Ove90, Ove91]. Une manière simple et générale de les définir, proposée dans [PB88], consiste à voir chaque articulation comme une ou plusieurs contraintes géométriques à maintenir au cours du temps entre les solides concernés. Ce formalisme permet de choisir le nombre de degrés de libertés de chaque articulation, et de les munir éventuellement de butées. L'exemple très simple d'une contrainte "point à point" entre deux solides, qui crée une articulation de type rotule (trois degrés de liberté en rotation et aucun en translation), est illustré par la figure 2.1. Ce formalisme mis en place, l'objet de l'étude consiste à concevoir un bon algorithme de simulation sous contraintes.

Le paragraphe qui suit présente la méthode que nous avons introduite pour maintenir ce type de contrainte, et qui a fait l'objet de deux publica-

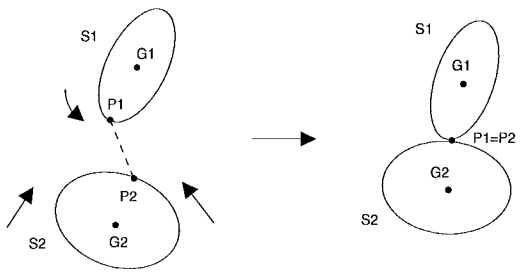


FIG. 2.1 - *Contrainte "point à point" entre deux solides.*

tions [GG92, GG94] dont l'une est inclue page 78 de ce mémoire. Au lieu de chercher à déterminer des forces de liaison qui maintiendraient les contraintes, ce qui conduit à l'intégration numérique d'un système d'équations différentielles couplées [BB88], nous avons opté pour une résolution en deux phases :

1. Animation individuelle des solides, comme s'ils étaient indépendants.

2. Maintien des contraintes grâce à une série de petites corrections en déplacement, ce qui est plus direct qu'un passage par l'intégration de forces.

3. Modification des vitesses en fonction des positions effectivement atteintes pendant le pas de calcul.

Le fait d'opter pour des corrections en déplacement rend la méthode utilisable pour une application de type "modélisation sous contraintes", pour laquelle on ne dispose pas de modèle physique des objets. Dans le cadre de l'animation par modèles générateurs, la cohérence dynamique du résultat est assurée par la conservation du moment du système des solides contraints, lors de la phase de résolution des contraintes. Détaillons ce processus.

Maintien d'une contrainte "point à point"

Maintenir une contrainte entre deux solides grâce à de petites corrections en déplacement admet une infinité de solutions. Cependant, pour que le mouvement obtenu soit physiquement crédible, la loi de l'action et de la réaction doit être respectée, en ce sens que c'est le solide le plus léger qui se déplacera le plus (conservation des moments de translation de premier ordre). De plus, un bon choix doit être fait pour la proportion entre translation et rotation du mouvement de chaque solide.

Pour déterminer ces proportions, qui dépendent des masses et inerties des deux solides, nous faisons l'analogie suivante : supposons que les points contraints des solides soient reliés par un ressort idéal, de longueur au repos nulle (voir la figure 2.1, où le ressort est représenté par la ligne pointillée). Calculer les rotations et translations produites par ce ressort est aisé. Nous en déduisons pour chaque objet une rotation et une translation permettant de vérifier la contrainte.

Contraintes multiples sur un solide

Lorsque plusieurs contraintes s'appliquent simultanément au même objet, leurs effets doivent être combinés, mais de manière à d'une part préserver les lois d'action et de réaction sur chaque couple de solides contraints, et à d'autre part ne pas introduire de divergence. Comme le montre [GG94], sommer directement les déplacements calculés par chaque contrainte respecte le premier pré-requis, mais peut faire diverger l'algorithme. Nous proposons l'emploi d'une combinaison linéaire des déplacements dont les coefficients sont fonction du nombre total de contraintes sur les solides concernés. La convergence a pu être démontrée dans un cas simple (contraintes ne donnant lieu qu'à des translations).

Autres types de contraintes

La méthode précédente se généralise facilement à des contraintes offrant des degrés de liberté en translation, éventuellement munis de butées (contraintes "point à segment", "point à plan", "point à anneau", "point à boîte", etc). Les trois degrés de liberté en rotation peuvent être réduits par l'ajout de contraintes d'angle entre différents vecteurs définis dans les repères locaux des solides. Ces extensions, qui fonctionnent toutes en calculant un couple de points des objets à placer au même point géométrique, sont détaillées dans [GG94].

Les test pratiques de la méthode nous ont permis de construire et d'animer des structures articulées pouvant comporter des boucles fermées, et cela avec un bon taux de convergence en moyenne (de l'ordre de 10 itérations pour la phase de traitement des contraintes).

Chapitre 3

Structure externe : matériau élastique implicite

La structure interne présentée au chapitre précédent permet de synthétiser les mouvements les plus complexes, mais ne suffit pas à la description d'un objet déformable, dans la mesure où aucune information géométrique n'est donnée. Nous avons déjà insisté sur le fait qu'une description surfacique précise des objets et de leurs déformations est primordiale pour les applications à la synthèse d'images et à la production, dans la mesure où ces surfaces seront utilisées pour le rendu de l'animation.

Ce chapitre, qui constitue le point central de ce travail d'habilitation, présente un modèle déformable original basé sur les fonctions implicites. Particulièrement compact, ce modèle permet de décrire de manière unifiée la géométrie des objets et la manière dont leur surface répond à une action extérieure (correspondance entre force appliquée et déformation). Utilisé comme un "composant externe" dont la position est à chaque instant définie par la structure interne[1], ce modèle joue le rôle d'interface avec le monde extérieur : il détecte les collisions avec les autres objets de la scène, se déforme localement, et transmet les forces de réaction à la structure interne. Cette dernière les prendra en compte au pas de temps suivant dans le calcul du mouvement. La particularité de la formulation implicite proposée est de permettre la définition de surfaces de contact mathématiquement exactes entre corps déformables en intéraction. Les équations de ces surfaces seront utilisées directement pour un rendu de qualité (lancer de rayons direct [Gas95]), ce qui assure la cohérence entre le modèle déformable et sa représentation finale.

Une description détaillée du modèle implicite déformable sera trouvée dans [Gas93], inclus page 93 de ce mémoire. Le modèle est alors combiné avec une structure interne de type "solide rigide", ce qui permet de modéliser des objets visco-élastiques pouvant subir des déformations locales, puis revenir à leur

1. Nous nous plaçons ainsi dans le cadre d'une architecture hybride, où le calcul du mouvement (composant interne) et des déformations locales (composant externe) sont séparés. Cela revient à négliger les variations du tenseur d'inertie dues aux déformations locales lors du calcul du mouvement. Ce type d'approximation a déjà été mis en œuvre avec des résultats satisfaisants dans plusieurs travaux antérieurs [TW88, TF88, GVP91].

forme initiale. Plus récemment, nous avons étendu le modèle initial au cas d'une structure interne déformable (voir [DG94], inclus page 102 de ce mémoire), dans le cadre de la synthèse de substances à grands champs de déformation. Pour cette application, une solution simple et efficace a du être trouvée pour l'échantillonnage de surfaces implicites pouvant changer de de topologie (fractures, fusions). Cette solution est détaillé dans [DTG95], inclus à la page 117. La présentation ci-dessous synthétise donc les principales idées mises en œuvre dans ces trois articles, ainsi que leurs prolongements récents. En particulier, nous expérimentons actuellement un module de contrôle permettant de réduire les variations de volume de l'objet lors des déformations de sa structure interne, et nous cherchons d'autre part à rendre possibles les inter-collisions entre les différentes parties d'une surface implicite.

3.1 Une représentation implicite de l'élasticité

Surfaces implicites isopotentielles

Le modèle s'appuie sur les surfaces implicites construites sous la forme d'iso-surfaces engendrées par des "squelettes" [WMW86, BW90]. Chaque squelette, qui peut être n'importe quelle primitive géométrique admettant une fonction de distance, engendre un potentiel qui décroit avec cette dernière (voir figure 3.1). La surface est définie en sommant les contributions de chaque squelette :

$$S = \{P \in I\!R^3 \ / \ f(P) = 1\} \ \text{ où } \ f(P) = \sum_{i=1}^{n} f_i(P)$$

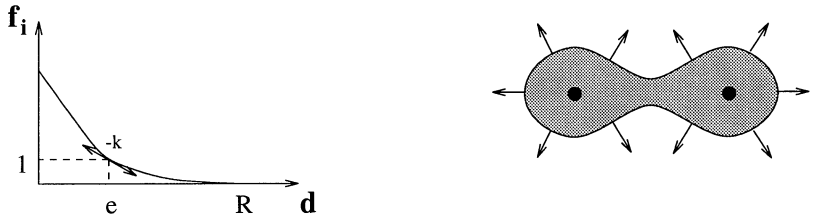


FIG. 3.1 - *Exemple de fonction potentiel, et surface engendrée par deux points.*

Les déformation locales ou globales peuvent par exemple s'exprimer par l'ajout d'un nouveau terme dans la fonction potentiel. L'équation de la surface devient alors $f(P) + g(P) = 1$, où g est le terme de déformation. Notons qu'un terme positif produit une dilatation, et un terme négatif une compression de l'objet.

Expression de l'élasticité

La forme d'une surface implicite est uniquement définie par l'ensemble des points où la fonction potentiel vaut 1. Les variations de cette fonction autour de la surface n'ont donc aucune influence sur la géométrie, et peuvent être utilisées pour stocker d'autres propriétés. L'idée de base du modèle consiste à choisir la

norme du gradient de la fonction potentiel en fonction de la raideur k que l'on désire donner à l'objet :

$$Grad(f, Y) = k(Y)\,\vec{n}(Y)$$

Soit $dR(Y)$ une force infinitésimale appliquée au point de coordonnée Y d'un solide. Si on exprime l'élasticité, qui nous dit que $dR(Y) = k(Y)dY$, en terme de variation de la fonction potentiel f définissant la géométrie de l'objet, on obtient alors :

$$d\,f(Y) = Grad(f, Y)\,.\,dY = Grad(f, Y)\,.\,\frac{dR(Y)}{k(Y)} = dR(Y)\,\vec{n}(Y)$$

Le potentiel de déformation g, obtenu par intégration des variations de f, s'exprime donc très simplement pour des déformations normales (pour lesquelles $\vec{n}(Y)$ reste constant). En chaque point de la surface :

$$g = -\vec{n}\,.\,R$$

Dans la pratique, nous utiliserons cette formule pour évaluer la force de compression radiale correspondant à un potentiel de déformation donné.

3.2 Traitement des contacts entre objets

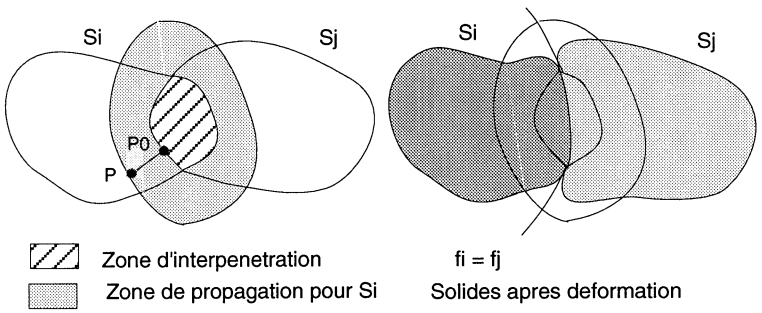


FIG. 3.2 - *Modéliser le contact consiste à appliquer différents potentiels de déformation dans la zone d'interpénétration et dans la zone de propagation des déformations.*

La principale contribution du modèle élastique implicite qui vient d'être décrit est d'engendrer des surfaces de contact exactes, de forme libre, entre corps déformables en contact. Pour cela, le traitement des interactions s'effectue en trois phases :

1. **Détection des interpénétrations :** Après une pré-détection utilisant des boites englobantes, nous tirons profit de l'expression implicite des objets pour accélérer le processus : un seul des objets est discrétisé et ceux de ses points qui appartiennent ŝa la boite englobante du second objet sont testés sur la fonction potentiel de ce dernier, ce qui indique si le point est intérieur ou extérieur.

2. **Modélisation du contact :** Il s'agit ici d'ajouter des termes de déformation aux potentiels des deux objets (voir figure 3.2) de manière à :

   - Engendrer des surfaces de contact exactes dans la zone d'interpénétration. Ces surfaces doivent correspondre à des forces de compression opposées appliquées aux deux objets.
   - Modéliser la propagation transversale des déformations.

   Dans la zone de contact, le potentiel de déformation à appliquer à S_i, peut être défini par $g_{ij}(P) = (1 - f_j(P))$, ce qui donne les résultats attendus. Le potentiel de propagation, dont le choix n'affecte pas le mouvement ultérieur des solides, est calculé de manière purement géométrique, et permet de conserver le caractère G^1 (continuité géométrique d'ordre 1) des surfaces (voir [Gas93]).

3. **Réponse aux collisions :** Les forces de réaction sont intégrées numériquement sur les points d'échantillonnage qui ont servi à détecter les collisions. Outre la force de compression radiale calculée en multipliant le potentiel de déformation en un point par l'aire de l'élément de surface qui lui est associé, ces forces comprennent les frottements visqueux liés à la différence de vitesse des deux objets aux points de contact.

   Les forces ainsi que les couples qu'elles engendrent sont communiquées à la structure interne de l'objet pour être prises en compte au pas de temps suivant.

Notons que cet algorithme est utilisé avec un intervalle de temps auto-adaptatif. En effet, un pas trop important pourrait produire de trop fortes interpénétrations entre les objets, qui rebondiraient alors plus vite qu'ils ne sont venus. Le problème est détecté et le simulateur reprend la simulation au début de l'itération annulée, avec un pas de temps deux fois moindre. L'intervalle de temps est augmenté à nouveau lorsqu'aucune exception n'est produite pour plusieurs itérations consécutives.

Cet algorithme peut être simplifié dans le cas où l'un des objets est considéré comme étant idéalement rigide. Une seule déformation est alors calculée. La méthode donne également des résultats exacts dans le cas de collisions multiples. Enfin, l'implémentation proposée dans [Gas93], qui correspond au cas d'objets élastiques ne subissant que des déformations locales, est basée sur un précalcul des points d'échantillonage avant l'animation (rappelons que ces points sont nécessaires pour la détection des collisions et pour l'intégration numérique des forces de contact). Les points d'échantillonage, stoqués dans le repère local de la structure interne, sont repositionnés à chaque instant sur la surface déformée. Cette approche permet un calcul efficace de l'échantilonage, dans la mesure où elle tire profit de la cohérence temporelle d'une animation.

3.3 Adaptation à une structure interne déformable

Comme nous l'avons vu au chapitre 2, les composants internes de base, points et solides, peuvent être connectés entre eux de manière à former des

réseaux déformables. Recouvrir ces structures complexes d'une couche de matériau élastique implicite permet de définir une surface lisse, capable de suivre les déformations et même les changements de topologie de la structure interne, de détecter les collisions, et de transmettre les forces de réaction calculées le long des surfaces de contact. Cela peut être fait très simplement, en définissant les squelettes qui engendrent la surface implicite dans les repères locaux des différent composants internes. Deux types d'applications en découlent :

Modélisation d'objets déformables structurés : Un personnage par exemple, constitué d'un squelette rigide articulé recouvert de chair déformable, peut être réalisé en définissant chaque élément de squelette (au sens des fonctions implicites) dans le repère local de l'un des composants d'une structure interne articluée (les structures articulées, constituées de solides reliés entre eux par des contraintes géométriques, ont été définies au chapitre 2).

Modélisation de substances malléables : Un bloc de matière à grands champs de déformations, pouvant subir des fractures, sera construit en définissant chaque squelette (au sens des fonctions implicites) comme étant confondu avec l'un des points matériels d'un système de particules. Ici, l'emploi d'un modèle implicite est particulièrement bien adapté, puisque les changements de topologie au cours des déformations du système interne se répercuteront automatiquement sur la surface de l'objet, qui pourra en particulier subir des fractures et des fusions.

Cependant, ces applications imposent une adaptation du matériau implicite décrit ci-dessus :

- L'échantillonnage de chaque objet ne peut plus être pré-calculé, comme c'était le cas dans [Gas93]. Il doit en effet s'adapter aux changements de topologie qui peuvent se produire au cours de l'animation, et qui demander l'ajout ou la suppression de points. Pour conserver l'interactivité, chaque nouvel échantillonage doit être engendré de manière très efficace.

- Lorsque la structure interne se déforme, le déplacement relatif des squelettes qui définissent la surface implicite peut conduire à d'importantes variations de volume. Il est primordial de pouvoir contrôler ces variations pour rendre l'animation plus crédible.

- Enfin, la stratégie de "mélange" de potentiels entre les différentes parties d'un objet doit être affinée. Par exemple, pour un modèle de personnage, rapprocher une main de l'une des jambes devrait produire un contact, et non une fusion des deux surfaces, bien qu'elles appartiennent au même objet. De même, deux blocs détachés d'une même substance malléable devraient entrer en collision, au lieu de fusionner à distance.

La suite de cette section passe en revue les solutions que nous apportons à ces trois problèmes.

Échantillonnage adaptatif efficace et optimisations

Nous avons vu qu'il est nécessaire de disposer de points d'échantillonnage à la surface des objets pour la détection des collisions d'une part, et pour l'intégration numérique des forces de réaction d'autre part. Ces points peuvent aussi servir à l'affichage interactif des objets en cours de calcul (on se reportera à [DTG95], inclus page 117 de ce mémoire, pour une description plus détaillées de notre méthode d'échantillonnage, et une discussion des modes d'affichage interactif qu'elle permet). Comme nous l'avons dit plus haut, un pré-échantillonnage ne convient plus pour des objets pouvant subir des formes variations de forme ainsi que des changements de topoilogie au cours de l'animation. Nous proposons donc une méthode d'échantillonnage adaptatif, que nous avons cherchée à rendre la plus efficace que possible.

L'idée de base est la suivante. Chaque squelette lance un certain nombre de "graines" (points géométriques) dans un ensemble de directions (appelées "axes") bien réparties et fixes par rapport à son repère local. Certaines de ces graines, qui entrent dans une zone où l'influence d'un autre squelette est prédominante, sont invalidées avant d'atteindre la surface. Les autres, appelées "graines valides", servent à son échantillonnage (voir figure 3.3). A chaque instant, les positions de toutes les graines le long de leur axe sont réactualisées, grâce à une dichotomie qui prend pour point de départ leur position précédente dans le repère local du squelette associé. En particulier, certaines graines invalides peuvent être validées de nouveau, et vice-versa. Cet algorithme permet d'exploiter la cohérence temporelle d'une animation (il n'y a aucun ajustement à faire dans les zones non déformées) tout en offrant une structure qui s'auto-adapte aux changements de topologie de la surface. Moins ambitieuse que la technique présentée dans [WH94] quant à la répartition des échantillons, cette méthode simple est efficace et bien suffisante pour notre application (voir [DTG95]).

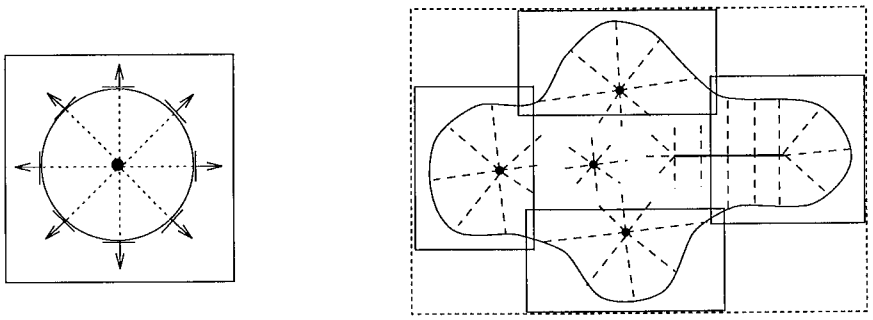


FIG. 3.3 - *Migration des graines pour un échantillonnage adaptatif.*

Utiliser une boîte englobante contenant tout l'objet n'a plus grand sens lorsque celui-ci s'est fracturé en plusieurs composantes connexes. Notre méthode d'échantillonnage permet de résoudre ce problème par la construction d'une hiérarchie de boîtes parallèles aux axes. Pour cela, une boîte est calculée pour chaque squelette à partir des positions de ses graines valides à l'instant courant. Chaque boîte est agrandie de la distance maximale entre deux échantillons, de

manière à ce que leur union constitue un pavage de la surface implicite (mais non nécessairement un recouvrement du volume qu'elle délimite, voir figure 3.3). Les boîtes associées aux squelettes sont regroupées dans une boîte qui englobe toute la surface, formant ainsi une hiérachie à deux niveaux qui permet d'optimiser les détections de collisions. En effet, lorsqu'une interpénétration entre deux boîtes mères a été détectée, l'utilisation des boites locales qui sont beaucoup plus proches des surfaces des objets permet d'éviter de nombreuses évaluations de potentiel, en particulier dans le cas d'objets voisins qui ne s'intersectent pas.

Enfin, les graines qui échantillonnent la surface sont utilisées pour le calcul des forces de réponse aux collisions. La structure interne étant maintenant déformable, constituée de plusieurs points ou solides mobiles les uns par rapport aux autres, un choix doit être fait concernant la répartition des forces entre ces différents éléments. Dans le cas des forces de compression, calculées sur une surface de contact dont les graines constituent un bon échantillonnage, nous nous contentons de transmettre la force calculée en chaque graine au squelette qui l'a envoyée (qui est aussi celui d'influence maximale en ce point). Cette méthode a l'avantage de réduire les calculs, et donne des résultats suffisants en pratique. Pour une force ponctuelle, une meilleure solution consisterait à répartir la force entre tous les squelettes qui contribuent au potentiel en ce point.

Le problème des variations de volume

Lorsque la structure interne se déforme, voire change totalement de topologie, rien n'assure que le volume global de l'objet soit conservé. Ceci est illustré par la Figure 3.4, dans le cas très simple de 4 particules se rassemblant en un même point : lorsqu'il n'y a aucune condition sur les fonctions potentiel, de fortes variations du volume implicite peuvent se produire au cours des déformations.

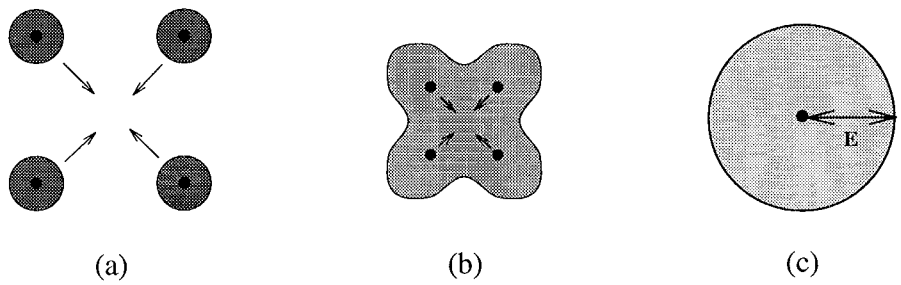


FIG. 3.4 - *Une compression de la structure interne de ne devrait pas augmenter le volume !*

Brian Wyvill [WMW86] avait déjà évoqué ce problème dans le cas particulier de deux squelettes ponctuels se rassemblant en un seul point. Il conservait le volume entre les configurations initiale et finale par le biais d'un choix judicieux de l'isopotentielle. Nous avons exprimé ce problème dans la situation un peu plus générale où un nombre quelconque de squelettes ponctuels se rassemblent en un seul point. Les contraintes qui en résultent sur la fonction potentiel (même volume dans les configurations initiale et finale de la figure 3.4) montrent que la

fonction $f(x) = (x_0/x)^3$ est solution, où x_0 est l'épaisseur de matériau autour d'un squelette lorsque ce dernier est isolé (voir [DG94]). Ce choix permet de limiter les variations de volume lorsque la structure interne reste proche d'un état d'équilibre.

Toutefois, la solution que nous venons d'évoquer n'est pas totalement satisfaisante, si bien qu'une méthode très différente est actuellement à l'étude. En effet, choisir une équation donnée pour la fonction potentiel est très limitatif, dans la mesure où cette fonction contrôle déjà à la fois la forme et la raideur de l'objet. De plus, l'utilisation de $f(x) = (x_0/x)^3$ laisse persister d'importantes variations de volume (jusqu'à 40% sur les exemples étudiés) lors de fractures ou de fusions. Nous nous penchons donc actuellement sur une méthode locale pour le contrôle de volume, basée sur l'utilisation de contrôleurs proportionnels intégraux dérivés (PID). L'idée de base consiste à attacher un contrôleur à chaque squelette, ce qui lui permet de réguler son potentiel en fonction des variations locales de volume (un volume local associé au squelette est calculé de manière approchée grâce aux positions des graines, valides ou non, lancées par le squelette). Les résultats préliminaires sont très prometteurs : nous parvenons pour l'instant à réduire les variations de volume à moins de 5%, même dans le cas de changements de topologie.

Mélange indésirable

Lorsque le composant interne d'un objet est déformable, mais reste structuré (c'est le cas pour un squelette articulé enrobé de matériau déformable par exemple), une définition plus fine de la notion de "mélange" de potentiels est nécessaire. Prenons l'exemple du personnage de la figure 3.5. Si un bras se rapproche du corps, les potentiels engendrés par ces deux parties de l'objet ne doivent pas se mélanger, bien qu'ils servent à la définition de la même surface implicite. En particulier, des inter-collisions entre une main et le torse du personnage doivent être détectées. Cependant, la distinction entre bras et corps ne peut pas être faite complètement, dans la mesure à ces deux objets n'en forment qu'un (mélange de potentiels au niveau de l'épaule).

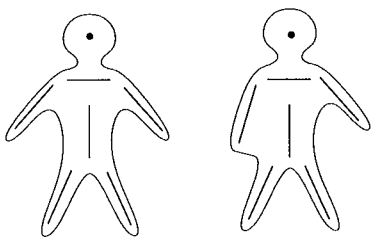


FIG. 3.5 - *Mélange indésirable sur un modèle de personnage.*

Ce problème, que nous appellerons ici "mélange indésirable" [2], peut être résolu par l'emploi d'un "graphe de mélange" associé à l'objet implicite. Ce graphe, fixe au cours du temps dans le cas d'une squelette articulé, indique à chaque squelette quels sont les "voisins" avec lesquels il peut mélanger son

2. Le terme consacré en anglais est "unwanted blending".

potentiel. Le calcul du potentiel en un point P est alors effectué de la manière suivante :

1. On détermine la contribution $f_i(P)$ qui est maximale parmi celles de tous les squelettes de l'objet :
2. On lui ajoute les contributions des "voisins" (au sens du graphe) du squelette S_i.

Comme nous le montrons dans [DTG95], inclus page 117, l'échantillonnage d'une surface implicite à l'aide des "graines" que nous avons définies est particulièrement bien adapté à la résolution de ce problème. En effet, chaque graine connaît son squelette d'influence maximale, qui est celui qui l'a lancé (la graine devenant invalide dès que cette contrubution n'est plus prépondérante), donc le calcul du potentiel se fait par simple ajout des contributions des squelettes voisins.

Notons qu'un problème analogue, mais de difficulté supérieure, se pose dans le cas de la simulation d'un matériau à grands champs de déformation, qui se fracture en plusieurs composants. En effet, lorsque ces derniers se rapprochent de nouveau, leur fusion ne doit pas s'effectuer à distance, mais uniquement s'ils sont compressés suffisamment fort l'un contre l'autre. Il s'agit dont ici encore de redéfinir les relations de voisinages, mais au sein d'une structure adaptative qui dépend des composantes connexes courantes du volume implicite. Une solution à ce problème est actuellement à l'étude.

Chapitre 4

Combiner simulation et interface type "positions clés"

4.1 Le modèle générateur comme outil d'aide au réalisme

Comme nous l'avons dit en introduction, l'un des principaux facteurs limitant l'emploi des modèles générateurs dans les logiciels d'animation du commerce est la difficulté de contrôle qui leur est inhérente. En effet, un animateur désire le plus souvent que les objets, ou du moins certains d'entre eux, se conforment à un scénario donné. S'il ne peut pas trouver facilement les forces ou les couples à leur appliquer pour que ce soit le cas, l'utilisateur peut par contre définir très facilement une ébauche de la trajectoire désirée sous forme de positions clés.

La méthode que nous présentons dans ce chapitre est destinée à rendre le modèle générateur plus accessible comme outil d'aide au réalisme. La création d'une animation se passe alors en deux temps : l'utilisateur commence par définir une ébauche de trajectoire pour tous les objets ou pour une partie d'entre eux, munis d'effecteurs, c'est à dire de dispositifs capables d'engendrer des forces et/ou des couples. Une phase de simulation permet alors de calculer l'action des effecteurs au cours du temps de manière à ce que le mouvement effectif des objets se rapproche de la trajectoire souhaitée. L'intérêt de la simulation est d'améliorer le mouvement pré-défini : les masses et inerties des objets sont prises en compte, ainsi que les déviations dues aux collisions éventuelles qui sont automatiquement détectées. Les variations de vitesse de chaque objet le long de sa trajectoire sont également déterminées par les calculs, et varient suivant la complexité de la trajectoire, la puissance des effecteurs, et les interactions éventuelles avec d'autres objets. Ce travail a fait l'objet de deux publications [LG93, LGG95], dont l'une est inclue page 130 de ce mémoire.

Une extension importante pour la mise au point de séquences d'animation complexes est de pouvoir spécifier des contraintes de synchronisation entre les mouvements des différents objets, permettant ainsi la définition de scénarios. Présentée brièvement dans la dernière partie de ce chapitre, cette extension fait l'objet de l'article [LG95], que l'on trouvera à la page 153 de ce mémoire.

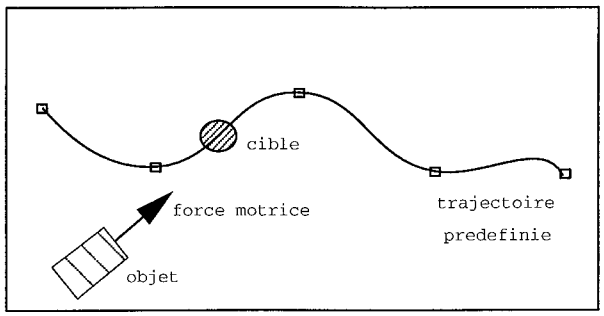


FIG. 4.1 - *Un objet guidé par une position "cible" sur la trajectoire associée.*

Notons que l'ensemble de ce travail fait partie de la thèse d'Alexis Lamouret, effectuée sous ma co-direction, et qui doit être soutenue au mois de juin 1995.

4.2 Guider la trajectoire des objets

Cas d'un objet isolé

Considérons un objet muni d'un effecteur, et dont a été spécifié une trajectoire approximative, en translation ou en rotation. Comment déterminer l'action de l'effecteur de manière à ce que l'objet, supposé isolé, suive cette trajectoire avec une précision spécifiée à l'avance?

Notre approche se base sur une technique de "course poursuite" entre l'objet et une position "cible" mobile sur la trajectoire, et dont l'abscisse curviligne ne peut que croître au cours du temps (voir figure 4.1). A chaque instant :

1. La force (ou couple) de l'effecteur est calculée en fonction des données cinématiques et dynamiques (position, vitesse, massse, inertie de l'objet) de manière à ce que ce dernier parcourt une proportion donnée de la distance à sa cible pendant l'intervalle de temps de simulation ;

2. La cible se déplace de manière à rester toujours à une distance D constante de l'objet.

Cette distance représente le seuil de précision avec lequel la trajectoire ébauchée par l'animateur sera suivie. Par la suite, lorsque l'objet sera dévié de sa trajectoire par une collision brutale, la cible attendra qu'il revienne à une distance inférieure à D pour poursuive son chemin.

La méthode de calcul des forces et couples des effecteurs est une généralisation de techniques de contrôle classiques en automatique (contrôleur proportionnel dérivé). Pour un effecteur en translation, positionné sur une objet de masse m, de position x, et de vitesse $\dot{x}$, la force produite est :

$$\vec{F}(t) = 2m \frac{\alpha \left(\overrightarrow{x_{\text{cible}}} - \vec{x}(t) \right)}{\Delta t^2} - 2m \frac{\beta \dot{x}(t)}{\Delta t}$$

Les paramètres α et β permettent de régler la vitesse de déplacement de l'objet, ainsi que la régularité et le caractère plus ou moins amorti de la trajectoire.

Leurs effets sont détaillés dans [LGG95] (page 130). Le même type de calcul est mis en œuvre pour trouver le couple produit par un effecteur de rotation.

Objets non-isolés, et structures complexes

Dans le cas général, un effecteur ne sera pas attaché à un corps isolé, mais à un objet soumis à son poids, en interaction éventuelle avec d'autres objets de la scène, et pouvant être l'un des composants d'une structure complexe (réseau d'interactions ou réseau de contraintes). Le module de contrôle doit alors prendre en compte l'effet des différentes actions extérieures appliquées à chaque instant à l'objet. Un nouveau terme, qui s'oppose à ces actions, est ainsi ajouté à la force donnée par la formule précédente.

De manière à ce qu'une collision brutale donne toujours lieu à une déviation de trajectoire, mais que celle-ci ne soit pas affectée par l'action d'une force constante, les actions extérieures sont prises en compte avec un retard. Comme nous ne disposons pas toujours de leur valeur (en particulier en ce qui concerne les contraintes, voir section 2.3), ces forces sont réévaluées en fonction des déviations ide trajectoire qu'elles produisent. Nous utilisons un filtrage temporel sur la force totale produite par chaque effecteur de manière à régulariser le mouvement, ce qui est particulièrement utile lorsque des contraintes sont en jeu.

Rappelons que l'intervalle de temps utilisé pour le calcul d'une animation est adaptatif. En particulier, il peut décroître lors des collisions calculées par le composant externe[1] des objets. Nous avons donc validé notre méthode de contrôle dans le cas extrême d'un intervalle de temps variant aléatoirement. Les résultats, présentés dans [LGG95], montrent la stabilité de la trajectoire obtenue.

4.3 Problèmes de synchronisation

Le travail décrit ci-dessus permet de contrôler le mouvement d'un ou plusieurs composants internes de base, appartenant ou non au même objet. Le jeu des contraintes permettra par exemple à un composant muni d'un effecteur de tirer le reste d'une structure complexe.

Cependant, il reste à régler le problème de la synchronisation entre les effets des différents effecteurs. Lors de la réalisation d'un film d'animation, l'animateur a en effet besoin de pouvoir synchroniser les trajectoire de deux parties du même objet (un personnage qui marche), ou de deux objets différents (une poignée de main). Pour cela, chacun des effecteurs devra, tout en suivant sa trajectoire, tenter d'adapter sa vitesse en fonction de la distance qui le sépare de son lieu de rendez-vous et de la position atteinte par l'autre.

La méthode proposée dans [LG95] (page 153) permet de définir certains points des trajectoires désirées dans des référentiels locaux à d'autres objets en mouvement, et utilise un graphe à états finis pour stocker les "rendez-vous" temporels entre différents sous ensembles des objets (un rendez-vous est ici

1. Ou "structure externe".

défini comme la contrainte pour différents objets d'atteindre simultannément des positions spécifiées à l'avance sur leurs trajectoires respectives). Les états d'avancement des objets par rapport à un rendez-vous donné sont déterminés en fonction des positions cible courantes sur leurs trajectoires. Ces valeurs donnent lieu à un ajustement des vitesses, qui permet à tous les objets concernés par le rendez-vous d'"attendre" celui qui a été le plus retardé.

Chapitre 5

Objets déformables complexes : Quelques applications

5.1 Modèles pour l'animation de synthèse

La création de modèles pour la production de films de synthèse est l'application la plus immédiate des travaux synthétisés dans ce mémoire. En effet, comme nous l'avons vu dans le premier chapitre, la structuration modulaire qui a été adoptée est tout particulièrement destinée à simplifier la construction et le contrôle de modèles à animer. Ces derniers peuvent aller des objets inanimés, rigides ou déformables, à des modèles de personnages simples, dont le mouvement est contrôlé par l'intermédiaire des techniques décrites au chapitre précédent.

Modélisation géométrique des objets à animer

Comme nous l'avons vu, nous nous intéressons tout particulièrement à l'animation d'objets dont la surface est définie par une équation implicite, ce qui permet une gestion fine et précise des contacts entre objets.

Dans la mesure où les modeleurs classiques dont nous disposons au laboratoire *iMAGIS* se limitent aux surfaces paramétriques, nous avons mis au point notre propre logiciel de modélisation interactive à l'aide de surfaces implicites. Ce travail s'est effectué sous ma direction dans le cadre du projet de magistère de Nicolas Tsingos [TG94]. Pour rendre le système utilisable par un graphiste, nous avons cherché à fournir des outils de création, de manipulation et de modification interactives d'objets qui soient simples et intuitifs, et qui fonctionnent via une interface entièrement graphique.

Les principales fonctionnalités du modeleur sont :

- La manipulation interactive des squelettes et des objets qu'ils engendrent dans une ou plusieurs fenêtres 2D ou 3D. En particulier, les fonctionnalités comprennent l'ajout, la suppression, le positionnement, la déformation d'un squelette, ainsi que des opérations couper/coller.

- La modification interactive, grâce à une fenêtre de visualisation 2D, des fonctions potentiel engendrées par chaque squelette (changement de l'épaisseur, de la raideur d'un objet).

- La possibilité de choisir entre plusieurs modes de visualisation rapide à partir de points d'échantillonnage: une première possibilité est d'afficher des "écailles orientées" centrées en ces points, ce qui donne une bonne idée de la courbure de la surface; nous offrons également une possibilité d'affichage polygonal par morceaux, dont le rendu opaque s'avère particulièrement pratique lorsque de nombreux objets sont présents.

Les objets sont affichés en temps réel après toute déformation, grâce à la méthode d'échantillonnage adaptatif que nous avons déjà présentée au chapitre 3. Offrir cette visualisation simplifiée lors du fonctionnement interactif du modeleur n'altère en rien le rendu final des objets. En effet, ce sont les paramètres définissant les surfaces (squelettes, paramètres des fonctions potentiel) qui sont stoqués et non les points d'échantillonnage. Nous calculons le rendu final par lancer de rayon direct sur les surfaces implicites. Outre les fonctionnalités déjà citées, le modeleur comprend l'appel d'un lancer de rayons progressif (utilisation du logiciel interactif "raypaint" de C. Kolb, modifié afin de pouvoir afficher des objets implicites [Gas95]), qui donne très rapidement une idée de ce que sera le rendu final.

Exemples de structuration de modèles pour l'animation

Bien que beaucoup des techniques que nous avons présentées dans ce mémoire ne soient disponibles que depuis peu, ou même encore en cours de mise au point, plusieurs agencements des structures internes et externes ont déjà été testés au sein de l'équipe, dans le cadre d'applications diverses. Nous les décrivons dans les paragraphes qui suivent.

Simulation et contrôle d'objets visco-élastiques : Le film "Simply Implicit" [FGGL93] a été réalisé pour illustrer le modèle d'objets visco-élastiques de [Gas93] (page 93). L'objectif était de produire une animation plus vivante que de simples exemples de collisions. Le scénario et la conception des objets ont été réalisés pas Morgane Furio, graphiste à l'ENSAD (Ecole Nationale Supérieure des Arts Décoratifs, Paris). La scène représente une chambre d'enfants, où des jouets déformables sont posés sur une étagère. Un ballon fait tomber successivement tous les jouets de l'étagère, pour venir prendre leur place.

Pour cette application, chaque objet a été modélisé par deux couches : une structure interne simple régie par la mécanique du solide; une couche de matériau implicite élastique pour la structure externe. Le ballon a de plus été muni d'un effecteur en translation, offrant un contrôle de haut niveau sur sa trajectoire (voir chapitre 4). Le traitement des collisions et les corrections de trajectoire qui en résultent, le calcul des variations de vitesse, et celui de la rotation du ballon sur lui même due aux contacts et aux frottements, sont effectués par la simulation. Cela simplifie le travail de l'animateur et ajoute à la crédibilité du résultat.

Animation de matière hautement déformable : Comme nous l'avons vu au chapitre 3, les surfaces implicites sont très bien adaptées à la modélisation de matériaux hautement déformables (pâte semi-fluide), capables d'absorber les fortes déformations et de se casser éventuellement en plusieurs composants. De plus, ce champ d'investigation est intéressant dans la mesure où peu de modèles permettent de l'aborder. Les principales difficultés sont liées aux changements de topologie des objets, et au contrôle de leur volume. Nous avons expérimenté dans ce cadre l'emploi d'une structure interne discrète, constituée par un système à particules, contrôlant une structure externe implicite (les squelettes utilisés sont positionnés sur les particules). Ce modèle hybride présente de nombreux avantages. Simple et flexible, la structure interne particulaire est probablement le modèle le mieux adapté à la modélisation d'une grande variété de matériaux inélastiques, comme nous l'avons remarqué au chapitre 1. Ses déformations se répercutent de manière immédiate sur la surface de l'objet, dont la représentation implicite permet tous les changements de topologie nécessaires. Enfin, la structure externe épouse la forme des obstacles avec lesquels elle est en contact, et transmet les forces de réaction calculées le long de ces surfaces à la structure interne.

Les résultats obtenus dans [DG94] (inclus page 102), à l'occasion du stage de DEA de Mathieu Desbrun effectué sous ma direction, sont prometteurs. Nos travaux actuels portent sur un contrôle plus fin des variations de volume et sur une meilleure gestion de la fusion progressive de deux blocs issus du même objet.

Objets flexibles "à mémoire de forme" : Jean-Christophe Lombardo et Claude Puech [LP95] expérimentent actuellement dans le cadre de la thèse du premier un modèle hybride proche du précédent, mais pour lequel la structure interne particulaire classique est remplacée par un système à "particules orientées". Ces particules sont munies de lois d'interaction très particulières, qui permettent de stocker des informations telles que la courbure locale désirée et de faire tendre les positions relatives d'une particule et de ses voisines vers cette forme au repos. Recouvertes de matériau déformable, les particules orientées forment un squelette flexible discret, capable de retrouver sa forme initiale après toute déformation n'excédant pas un certain seuil. L'ensemble offre une représentation très compacte pour des objets déformables "à mémoire de forme".

Animation de personnages simplifiés : Enfin, l'un des essais les plus intéressants pour de futures applications à la production est l'animation de modèles simplifiés de personnages. Nous proposons de les modéliser en combinant une structure interne rigide articulée (solides reliés par des contraintes) et une structure externe déformable, représentant la "chair" du personnage et permettant de modéliser ses interactions avec l'extérieur. Ce type d'objet fait intervenir les techniques de "mélange indésirable" décrites au chapitre 3. Ainsi, des collisions et des contacts peuvent être calculés entre différentes parties d'un même objet. Un travail sur ce thème a débuté au laboratoire, dans le cadre d'une colla-

boration entre Mathieu Desbrun, qui débutera sa thèse sous ma direction en septembre prochain et Agata Opalach, chercheur invitée au laboratoire en septembre 1994 et candidate à une bourse post-doctorale à *iMAGIS* pour l'année universitaire 1994-1995.

5.2 Reconstruction semi-automatique d'organes

Collaboration avec l'équipe GMCAO (laboratoire TIMC/IMAG)

Les gestes médicaux assistés par ordinateur, qui consistent à aider médecins et chirurgiens à planifier et à réaliser des interventions grâce aux informations issues de l'imagerie, font l'objet d'un intérêt croissant de la communauté graphique depuis quelques années. Il s'agit en effet d'un domaine d'application très motivant, pour lequel les techniques récentes développées en images de synthèse (recherche du temps réel, gestion de la complexité) semblent d'un intérêt primordial.

Dans ce cadre, l'une des perspectives serait de construire un "atlas" type du corps humain, dont chaque organe de synthèse pourrait ensuite être recalibré très rapidement pour correspondre aux données scanner ou échographiques du patient à traiter. Une première étape consiste à se contenter d'une représentation géométrique des organes, qui pourra servir à la préparation d'une opération, à la fabrication de prothèses, ou à l'aide au positionnement des appareils au cours d'une intervention. A plus long terme, il serait très intéressant d'en proposer un modèle bio-mécanique, sur lequel l'intervention du médecin pourrait être simulée.

Depuis juillet 1994, une collaboration a été lancée sur ce thème avec l'équipe de Philippe Cinquin au laboratoire *TIMC/IMAG* de Grenoble. Il s'agit d'expérimenter l'utilisation des surfaces implicites, intéressantes pour leur caractère régularisant et très compact, pour reconstruire un organe type à partir de données médicales. Les participants du coté d'*iMAGIS* sont Nicolas Tsingos (dans le cadre de son projet de magistère) et moi-même. Nous travaillons avec Eric Bittar et Stéphane Lavallée du coté de *TIMC*.

Le travail réalisé jusqu'à présent fait l'objet d'un premier papier [TBG95], inclus page 165 de ce mémoire. La méthode proposée permet la reconstruction d'un organe sous forme de surfaces implicites à partir d'un nuage de points bruités. Le processus choisi est semi-automatique, de manière à ce que les connaissances déjà acquises par un spécialiste puissent être exploitées. L'utilisateur initialise un certain nombre de squelettes destinés à engendrer la surfaces implicite dans des zones particulières (ce qui permet d'obtenir un certain contrôle local). Une alternance entre des phases d'optimisation sur les positions et le potentiel des squelettes et des phases de fission automatique de certains de ces squelettes permet d'approcher les points mesurés avec un seuil de tolérance fixé au départ.

Ce premier volet de la collaboration devrait être suivi par l'étude de la calibration automatique d'un organe type sur les données d'un patient, puis par un travail sur la simulation de ses déformations.

5.3 Animation synchrone de "lèvres parlantes"

Collaboration avec l'Institut de la Communication Parlée (ICP)

La vision du visage du locuteur augmente remarquablement l'intelligibilité de la parole, surtout lorsque celle-ci est acoustiquement dégradée. Même modélisé par ses simples contours, le vermillon des lèvres apporte à lui seul environ un tiers de l'information visuelle transmise par un visage. L'*ICP* utilise actuellement pour ses expériences une modélisation purement géométrique des lèvres à partir de l'analyse de celles d'un locuteur. Or, cette modélisation présente des défauts dans les cas limites d'occlusion ou de forte protrusion.

Dans le cadre d'une collaboration avec l'équipe de Christian Benoit, de l'ICP, nous testons actuellement l'utilisation des "objets déformables complexes" pour la modélisation et l'animation de lèvres de synthèse. Ce travail fait l'objet du projet de DEA de Nicolas Tsingos, s'étendant sur l'année universitaire 1994-1995, et co-dirigé par Christian Benoit et par moi-même. La première étape consiste en une reconstruction semi-automatique des lèvres à partir d'une base de 23 formes de référence représentant le jeu des lèvres en Français. Une seconde étape consiste à reproduire la gestuelle de lèvres parlantes à partir de paramètres mesurés sur un locuteur. L'utilisation du modèle déformable implicite proposé dans ce mémoire devrait permettre un traitement simple des contacts entre les lèvres, et entre les lèvres et les dents.

Ultérieurement, ce travail pourrait déboucher sur un contrôle purement biomécanique des lèvres de synthèse, qui seraient intégrées à un modèle de visage complet.

5.4 Simulation de trajectographie de blocs pour la géologie

Collaboration avec le Bureau de Recherches Géologiques et Minières (BRGM)

Les mouvements de terrain recouvrent des phénomènes extrêmement variés, depuis la simple chute d'un bloc jusqu'à l'écroulement en masse (plusieurs millions de mètres cube). Le projet "Risque mouvements de terrain" du *BRGM*, dirigé par Eric Leroi à Marseille, a pour tâche d'étudier les risques de destruction encourus, et de travailler à leur prévention. Dans ce cadre, un modèle simple en deux dimensions pour la trajectographie de blocs a déjà été développé. Du fait de l'imperfection des connaissances de la géométrie du sol et de ses propriétés géo-mécaniques, ce modèle fait appel à une réponse aléatoire au niveau des impacts avec le sol. De plus, son extension directe à la dimension trois s'est avérée impossible. Parallèlement, des données expérimentales ont été recueillies sur le terrain, permettant d'étalonner plus finement le comportement déformable des sols.

Le *BRGM* a donc fait appel à nous pour aider à la conception d'un modèle 3D pour la trajectographie de blocs. Notre rôle est double. Il consiste d'une part à proposer des méthodes de reconstruction géométrique 3D des objets à animer en fonction de données mesurées sur le terrain, et d'autre part de résoudre

l'une des difficultés inhérentes à ce travail : l'explosion combinatoire liées au nombre très grand de blocs à simuler simultanément. Nous travaillons donc à proposer des algorithmes de détection des collisions particulièrement efficaces. En revanche, les modèles géo-mécaniques des objets à simuler seront fournis par le BRGM. Ce travail fait l'objet du stage de magistère de Frédéric Séraphine, effectué sous ma direction, et dont la seconde partie (juillet-août 95) aura lieu au BRGM à Marseille.

Chapitre 6

Bilan et perspectives

L'une des perspectives les plus prometteuses pour les systèmes d'animation de synthèse serait de permettre à l'animateur ou à l'artiste qui les manipule de consacrer pleinement son temps et son énergie au réglage très précis des mouvements et des déformations de premier plan, les autres objets de la scène étant contrôlés à un plus haut niveau d'abstraction. En déchargeant ainsi l'animateur d'une partie de son travail, de tels systèmes pourraient permettre d'aller vers la synthèse de scènes beaucoup plus complexes, faisant intervenir un très grand nombre d'objets ou de personnages en mouvement.

Dans ce cadre, la conception de modèles générateurs, capables d'engendrer mouvements et déformations de manière autonome est essentielle. Ces modèles, qui se chargeront en particulier de la gestion des interactions avec les autres objets de la scène, doivent rester simples à définir et à contrôler. Pour cela, deux aspects sont fondammentaux : le caractère intuitif des paramètres d'une part, et l'interactivité des calculs d'autre part, qui est essentielle pour les mouvements puissent être rapidement ajustés au vu des résultats.

Nous venons de faire la synthèse d'un certain nombre de contributions récentes pour la définition et le contrôle de nouveaux modèles générateurs. Elles comprennent en particulier des modèles de synthèse du mouvement pour des composants isolés ou bien connectés sous forme de structures complexes ("composants internes"), un modèle déformable original qui offre une gestion très précise des contacts entre objets ("structure externe"), et un module de contrôle de trajectoire destiné à être combiné avec les structures précédentes.

Il est clair qu'aucun des modèles que nous donnons en exemple ne peut prétendre à l'universalité, surtout dans la mesure où chaque module étudié a été choisi pour son caractère novateur plutôt que dans le but de définir un système d'animation complet. De manière à ne pas restreindre l'évantail des objets ou des comportements étudiés, chaque contribution présentée a été intégrée au sein d'une architecture modulaire, qui ne demande qu'à être enrichie. Des éléments plus classiques, comme par exemple un module de traitement des collisions et des contacts entre solides rigides, sont en cours d'intégration (pour cet exemple, il s'agit d'un nouveau type de "structure externe", en utilisant la terminologie proposée dans ce mémoire). Notons enfin que l'ajout de composants venant successivement enrichir un système modulaire permet de faire coexister et intéragir

facilement différents types d'objets, et de factoriser les efforts de chacun.

Le dernier élément de ce bilan porte sur la modélisation d'objets à l'aide de surfaces implicites isopotentielles. Ces surfaces, dont l'usage ne s'est répandu que très récemment en synthèse d'images, ont été introduites dans le cadre de ce travail d'habilitation pour faciliter la gestion des contacts entre objets déformables. Constituant l'une des premières tentatives pour associer modèle générateur et surfaces implicites, cette étude a eu de multiples retombées, dépassant de loin le cadre de l'animation par modèles générateurs. J'ai ainsi abordé les problèmes de la modélisation interactive à l'aide de surfaces implicites, celui de leur visualisation temps réel, la gestion des mélange indésirables de potentiel, ainsi que la reconstruction automatique d'objets à partir de points de données. Les applications, qui vont de la modélisation géométrique et de l'imagerie médicale à l'animation de synthèse, m'offrent actuellement de nombreuses perspectives de rencontres et de collaborations.

Bibliographie

[AG85] W. Armstrong and M. Green. The dynamics of articulated rigid bodies for purposes of animation. In *Graphics Interface 85*, pages 407–415, Montréal, May 1985.

[Arn94] Bruno Arnaldi. Modèles physiques pour l'animation. *Mémoire d'habilitation*, IRISA / INRIA et IFSIC, février 1994.

[Bar84] Alan H. Barr. Global and local deformations of solid primitives. *Computer Graphics*, 18(3):21–29, 1984.

[Bar89] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(3):223–232, July 1989.

[Bar90] David Baraff. Curved surfaces and coherence for non-penetrating rigid-body simulation. *Computer Graphics*, 24(4):19–28, August 1990. Proceedings of SIGGRAPH'90.

[Bar91] D. Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 25(4):31–40, July 1991.

[BB88] R. Barzel and A. Barr. A modeling system based on dynamic constraints. *Computer Graphics*, 22(4):179–188, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[BW90] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, March 1990.

[BW92] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics*, 26(2):303–308, July 1992. Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 1992).

[CHP89] J.E. Chadwick, D.R. Haumann, and R.E. Parent. Layered construction for deformable animated characters. *Computer Graphics*, 23(3):243–252, July 1989.

[Cia85] Ciarlet. *Elasticite Tridimensionnelle.* Masson, 1985.

[DAH89] G. Dumont, B. Arnaldi, and G. Hegron. Mechanics of solids for computer animation. In *PIXIM 89*, pages 293–308, Paris, October 1989.

[DG94] Mathieu Desbrun and Marie-Paule Gascuel. Highly deformable material for animation and collision processing. In *5th Eurographics Workshop on Animation and Simulation*, Oslo, Norway, September 1994.

[DTG95] Mathieu Desbrun, Nicolas Tsingos, and Marie-Paule Gascuel. Adaptive sampling of implicit surfaces for interactive modeling and animation. In *Implicit Surfaces'95*, Grenoble, France, April 1995.

[Dum90] G. Dumont. Animation de scènes tridimensionnelles : la mécanique des solides comme modèle de synthèse du mouvement. *Thèse de Doctorat*, Université de Rennes I, May 1990.

[FGGL93] Morgane Furio, Marie-Paule Gascuel, Jean-Dominique Gascuel, and Alexis Lamouret. Simly implicit. In *Film présenté dans la Session Papiers de la conférence SIGGRAPH 1993*, Anaheim, California, 1993, August 1993.

[For91] D.R. Forsey. A surface model for skeleton-based character animation. In *Second Eurographics Workshop on Animation and Simulation*, September 1991.

[Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics*, pages 313–320, August 1993. Proceedings of SIGGRAPH'93.

[Gas94] Jean-Dominique Gascuel. Fabule : un environnement de recherche pour l'animation et la simulation. In *Les Simulateurs, Troisième Séminaire du groupe de travail français Animation et Simulation*, October 1994.

[Gas95] Jean-Dominique Gascuel. Implicit patches: An optimized and powerful ray intersection algorithm for implicit surfaces. In *Implicit Surfaces'95*, Grenoble, France, April 1995.

[GG92] J.D. Gascuel and M.P. Gascuel. Displacement constraints: A new method for interactive dynamic animation of articulated solids. In *Third Eurographics Workshop on Animation and Simulation*, September 1992. To appear in The Visual Computer, 1993.

[GG94] M.P. Gascuel and J.D. Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer*, 10(4), March 1994.

[GMTT89] Jean-Paul Gourret, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of object and human skin deformations in a grasping task. *Computer Graphics*, 23(3):21–29, July 1989. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989).

[GP91] M.P. Gascuel and C. Puech. Dynamic animation of deformable bodies : State of the art report. In *Eurographics 91*, September 1991.

[GVP91] Marie-Paule Gascuel, Anne Verroust, and Claude Puech. A modeling system for complex deformable bodies suited to animation and collision processing. *Journal of Visualization and Computer Animation*, 2(3), August 1991. A shorter version of this paper appeared in *Graphics Interface'91.*

[IC87] P.M. Isaacs and M.F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics*, 21(4):215–224, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

[IC88] P.M. Isaacs and M.F. Cohen. Mixed method for complex kinematic constraints in dynamic figure animation. *The Visual Computer*, 2(4):296–305, December 1988.

[Jim93] Stéphane Jimenez. Modélisation et simulation physique d'objets volumiques déformables complexes. Thèse de doctorat, Institut National Polytechnique de Grenoble, November 1993.

[LC86] A. Luciani and C. Cadoz. Utilisation de modèles mécaniques et géométriques pour la synthèse et le contrôle d'images animées. In *Deuxième Colloque Image, CESTA*, Nice, April 1986.

[LG93] Alexis Lamouret and Marie-Paule Gascuel. An approach for guiding colliding physically-based models. In *Fourth Eurographics Animation and Simulation Workshop*, pages 209–219, September 1993.

[LG95] Alexis Lamouret and Marie-Paule Gascuel. Scripting interactive physically-based motions with relative paths and synchronization. In *Accepté à Graphics Interface'95*, Quebec, Canada, May 1995.

[LGG95] Alexis Lamouret, Marie-Paule Gascuel, and Jean-Dominique Gascuel. Combining physically-based simulation of colliding objects with trajectory control. *To appear in The Journal of Visualization and Computer Animation*, 1995.

[LJF+91] A. Luciani, S. Jimenez, J-L. Florens, C. Cadoz, and O. Raoult. Computational physics: a modeler simulator for animated physical objects. In *Eurographics'91*, Vienna, Austria, September 1991.

[LJR+91] Annie Luciani, Stéphane Jimenez, Olivier Raoult, Claude Cadoz, and Jean-Loup Florens. An unified view of multitude behaviour, flexibility, plasticity, and fractures: balls, bubbles and agglomerates. In *IFIP WG 5.10 Working Conference*, Tokyo, Japan, April 1991.

[LP95] Jean-Christophe Lombardo and Claude Puech. Oriented particles: A tool for shape memory objects modelling. In *Graphics Interface'95*, Quebec, Canada, May 1995.

[Luc85] A. Luciani. Un outil informatique de création d'images animées. *Thèse de docteur ingénieur d'électronique*, Institut National Polytechnique de Grenoble, November 1985.

[Mil88] Gavin Miller. The motion dynamics of snakes and worms. *Computer Graphics*, 22(4):169–177, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[MP89] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *SIGGRAPH '89 Courses 30 notes*, pages 305–309, August 89.

[MT91] Dimitri Metaxas and Demetri Terzopoulos. Constrained deformable superquadrics and nonrigid motion tracking. In *CVPR*, pages 337–343. IEEE Computer Society Conference, June 1991. Lahaina, Maui, Hawaii.

[MW88] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4):289–298, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[Ove90] C. Van Overveld. A technique for motion specification in computer animation. *The Visual Computer*, 6:106–116, 1990.

[Ove91] C. Van Overveld. An iterative approach to dynamic simulation of 3-D rigid-body motions for real-time interactive computer animation. *The Visual Computer*, 7:29–38, 1991.

[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.

[PTVF92] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C, second edition.* Cambridge University Press, New York, USA, 1992.

[PW89] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222, July 1989. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989).

[SP86] T.W. Sedeberg and S.R. Parry. Free-form deformations of solid geometric models. *Computer Graphics*, 20(4):151–160, 1986.

[TBG95] Nicolas Tsingos, Eric Bittar, and Marie-Paule Gascuel. Semi-automatic reconstruction of implicit surfaces for medical applications. In *Computer Graphics International'95*, Leeds, UK, jun 1995.

[TF88] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformations: Viscoelasticity, plasticity, fracture. *Computer Graphics*, 22(4):269–278, August 1988. Proceedings of SIGGRAPH'88.

[TG94] Nicolas Tsingos and Marie-Paule Gascuel. Un modeleur interactif d'objets définis par des surfaces implicites. In *Secondes Journées de l'AFIG*, Toulouse, December 1994.

[Ton91] D. Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface'91*, pages 255–262, Calgary, Canada, June 1991.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

[TPF89] Demetri Terzopoulos, John Platt, and Kurt Fleisher. Heating and melting deformable models (from goop to glop). In *Graphics Interface'89*, pages 219–226, London, Ontario, June 1989.

[TW88] Demetri Terzopoulos and Andrew Witkin. Physically based model with rigid and deformable components. *IEEE Computer Graphics and Applications*, pages 41–51, December 1988.

[WH94] Andrew Witkin and Paul Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics*, pages 269–278, July 1994. Proceedings of SIGGRAPH'94.

[WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, pages 227–234, August 1986.

[WW90] Andrew Witkin and William Welch. Fast animation and control for non-rigid structures. *Computer Graphics*, 24(4):243–252, August 1990. Proceedings of SIGGRAPH'90 (Dallas, Texas, August 1990).

Deuxième partie

Articles et Publications de 1991 à 1994

Chapitre 1

Dynamic animation of deformable bodies

Ce travail de synthèse sur les techniques d'animation d'objets déformables, qui date de 1991, a fait l'objet d'une heure de présentation dans le cadre des "State of the Art Reports" de la conférence internationale Eurographics, et a été sélectionné pour parution dans un ouvrage de référence :

[GP91] Dynamic Animation of Deformable Bodies : State of the Art Report. Marie-Paule Gascuel and Claude Puech. *State of the Art Reports of Eurographics 91*, Septembre 1991.

[GP94] Dynamic Animation of Deformable Bodies. Marie-Paule Gascuel and Claude Puech. *From Object Modelling to Advanced Visual Communication*, éditeurs S. Coquillart, W. Straber et P. Stucki, série *Focus on Computer Graphics*, pages 118–139, Springer-Verlag 1994.

Résumé

L'utilisation des lois de la mécanique est particulièrement bien adaptée à l'animation d'objets complexes comme les corps déformables, pour lesquels les techniques traditionnelles d'animation par "positions-clés" sont d'un usage difficile. Un avantage supplémentaire des modèles générateurs est de permettre une gestion automatique de la détection et de la réponse aux collisions. Cet article passe en revue les principaux modèles déformables existants, discute leurs avantages et limitations, et étudie les solutions qu'ils offrent au problème difficile du traitement des interactions entre objets.

S. Coquillart W. Straßer P. Stucki (Eds.)

From Object Modelling to Advanced Visual Communication

This book is a collection of the best papers originally presented as state-of-the-art reports or tutorials at the EUROGRAPHICS'91 conference in Vienna. A choice has been made giving priority to information of lasting value. Another goal was to cover all aspects of computer graphics - except hardware - as completely as possible from modelling to advanced visualization and communication. The editors consider that the ten contributions by internationally renowned experts fulfil this goal perfectly. Some important problem areas are treated from different viewpoints thus enhancing and deepening the reader's perspective.

Sabine Coquillart
Wolfgang Straßer
Peter Stucki

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

56 015

Dynamic Animation of Deformable Bodies

Marie-Paule Gascuel, Claude Puech

ABSTRACT
The use of mechanical laws is particularly convenient to animate complex objects such as non-rigid bodies, which would be very hard to imitate with key-frames. Moreover, dynamic models can provide natural answers to automatic collision detection and response. In this paper, we review the main existing models, discuss their advantages and limitations, and study the solutions they offer to the highly challenging problem of interactions.

Keywords:

Modeling, Animation, Deformation, Dynamics, Elasticity, Simulation, Collision detection, Collision response.

1 Introduction

To produce natural-looking animation sequences, the fact that each individual frame seems realistic is not sufficient. A coherent succession of images has to be produced in order to give the impression that the actors move and react as if they were part of the real world. They must comply with gravity, avoid interpenetrations, react in a natural way to collisions, and deform as the modeled material.

The classical key-frame animation systems are purely descriptive: the user specifies points of the prescribed trajectories, and controls interactively the possible interactions between objects. With this kinematic method, creating "realistic" animations is quite hard to achieve, particularly when some of the objects involved are non-rigid.

On the opposite, dynamic models generate movement and deformations of the objects according to their physical structure and to simplified physical laws. The movement is generally controlled through external actions (such as forces and torques) specified by the user. As emphasized in [25], the use of fundamental physical principles yields unsurpassed realism. Moreover, such "active models" can provide a natural solution to automatic interaction detection and response. In consequence, the use of dynamic laws to animate deformable bodies has been a topic of wide current interest during the past few years.

The remainder of the paper develops as follows:

Section 2 describes a first class of deformable models used in computer graphics. Based on the physical theory of elasticity in continuous media, these models offer very realistic results.

On the other hand, layered deformable models have been proposed. They integrate discrete mechanical components (such as masses, dampers and springs), which can be combined with purely geometric ones. Those models, listed in Section 3 can achieve interactive natural-looking animations of complex heterogeneous objects.

We present in Section 4 some optimization techniques that can be added to the previous approaches to specify constraints on the behavior of the objects.

As emphasized in Section 5, the automatic detection and response to interactions should be one of the main advantages of dynamic models, but remains a highly challenging problem. Indeed, the solutions advocated are not always generaly applicable; some of them introduce non-intuitive artificial mechanical elements during periods of contact.

Section 6 gives conclusions, and discusses the trends for future research, which include improved solutions for the interaction problem, and better control of the animation through hybrid models integrating dynamic and geometric specifications.

2 Deformable models based on the elasticity theory

This first class of models are derived from mathematical physics. The differential elasticity equations in continuous materials are discretized using finite elements or finite differences methods, and then integrated though time.

2.1 Terzopoulos, Platt, Barr and Fleischer's model

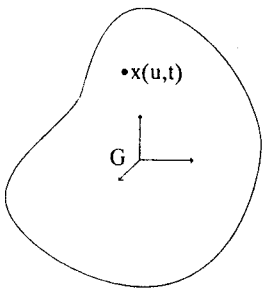


FIGURE 1. Elastically deformable body

The first model based on the elasticity theory was proposed in 1987 by Terzopoulos, Platt, Barr and Fleischer [23]. An elastically deformable body Ω (see figure 1), of parametrization $x(u,t) = (x_1(u,t), x_2(u,t), x_3(u,t))$ (u is a vector of coordinates (u_1, u_2, u_3) for a solid, (u_1, u_2) for a surface), is governed by Lagrange's ordinary differential equation:

$$\frac{\partial}{\partial t}\left(\rho\frac{\partial x}{\partial t}\right) + \gamma\frac{\partial x}{\partial t} + \frac{\partial \epsilon(x)}{\partial x} = f(x,t) \tag{1}$$

- $\frac{\partial}{\partial t}\left(\rho\frac{\partial x}{\partial t}\right)$ models the inertial force.
- $\gamma\frac{\partial x}{\partial t}$ represents the damping force due to dissipation.
- $\frac{\partial \epsilon(x)}{\partial x}$ is the elastic response of the material. It expresses the minimization through time of the functional $\epsilon(r)$, which measures the net instantaneous potential energy of the elastic deformation.
- $f(x,t)$ is the sum of the externally applied forces.

The animation is produced by integrating through time equation (1). More precisely, it involves the following computations:

1. The potential energy ϵ must be constant during a rigid motion, so a non-linear formula holds. Let G be the metric tensor associated with the object (this matrix is 3×3 for a solid, 2×2 for a surface). Let B be the curvature tensor if the body is an elastic surface (n is the unit normal vector at point $x(u,t)$ of the surface):

$$\begin{aligned} G_{ij}(x(u,t)) &= \frac{\partial x}{\partial u_i}\frac{\partial x}{\partial u_j} \\ B_{ij}(x(u,t)) &= n\,\frac{\partial^2 x}{\partial u_i\,\partial u_j} \end{aligned}$$

Then,

$$\begin{aligned} \epsilon(x) &= \int_\Omega \|G(r) - G^0(r)\|^2 du_1 du_2 du_3 \quad \text{for a deformable solid} \\ \epsilon(x) &= \int_\Omega \|G(r) - G^0(r)\|^2 + \|B(r) - B^0(r)\|^2 du_1 du_2 \quad \text{for a surface} \end{aligned}$$

Different types of elastically deformable materials can be obtained by modifying the matrix norm which is used. For instance, for an isotropic solid verifying Hooke's law[1]:

$$\epsilon(x) = \int_\Omega \left[\frac{1}{2}\lambda \left(\sum_i \left(G_{ii} - G^0_{ii}\right)\right)^2 + \mu \sum_{i,j} \left(G_{ij} - G^0_{ij}\right)^2\right] du \tag{2}$$

where λ and μ are Lamé constants for this material.

2. To compute $\frac{\partial \epsilon(x)}{\partial x}$, a formula from Calculus of Variations must be used:
If F is such as $\epsilon(x) = \int_\Omega F(u,x,x')\,du$, then:

$$\frac{\partial \epsilon(x)}{\partial x} = \frac{\partial F}{\partial x} - \frac{d}{du}\left(\frac{\partial F}{\partial x'}\right) \tag{3}$$

3. Then, the sum of the external forces must be evaluated (these forces can include gravity, friction, simulation of collision with rigid bodies, and users' actions).

4. Equation (1) is discretized in space, in order to obtain a system of coupled ordinary differential equations.

5. Finally, the previous system is integrated through time. At each time step, we obtain a linear system:

$$A_t\, x_{t+dt} = g_t \tag{4}$$

where A_t is a matrix representing the state of the deformable material at time t (the size of A_t is proportional to the number of reference points in the body), x_{t+dt} is the unknown vector-position, and g_t includes the sum of the external forces acting on the deformable body at time t. Equation (4) is solved using classical techniques.

This dynamic model yields realistic animation of various deformable materials, such as elastic clothes, sheets of paper, flexible metal bars, or rubber.

Nevertheless, it has several drawbacks:

- The system is not easy to handle, because it involves a large amount of computations. As a matter of fact, a precise control of the effects is difficult to achieve with a non-interactive system.

[1] Hooke's law states that the force on an object is linear in the displacement from a rest state.

- Global properties of the material (such as non-compressiveness for instance) cannot be specified.
- It is difficult to find a good spatial discretization if the shape of the object is non-trivial.
- The discrete equations become ill-conditioned when rigidity increases (see [25]). For instance, the movement of a piece of inextensible cloth could not be obtained (notably the apparition of folds).

2.2 Hybrid deformable model

As mentioned previously associating a nil deformation energy to a pure rigid motion gives of a non-linear formula for this energy. But such a model involves a great amount of computations, and the equations become ill-conditioned when rigidity increases. To cope with these problems, Terzopoulos and Witkin [25] propose an hybrid model combining rigid and deformable components.

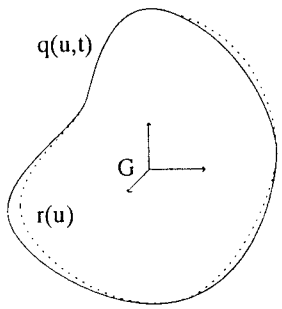


FIGURE 2. Hybrid deformable model with a rigid component

The body (which is parametrized by $q(u,t) = (q_1(u,t), q_2(u,t), q_3(u,t))$ in a referential linked to its center of mass) is divided in two interrelated levels (see figure 2):

$$q(u,t) = r(u) + e(u,t) \tag{5}$$

where :

- $r(u)$ is the rigid "reference component", which evolves according to the laws of rigid bodies dynamics.
- $e(u,t)$ is the "displacement component", which models the difference between the actual shape of the body and its reference shape. Globally still with respect to the reference component, $e(u,t)$ is animated with a linear formula for the deformation energy:

$$\epsilon(e) = \int E(\omega, e, e_u, e_{uu}....)du \tag{6}$$

  where the density of elastic energy E is a linear combination of the $e(u,t)$'s partial derivatives (see [25]).

After spatial discretization, and numerical integration through time, the linear system which must be solved at each time step if of the form:

$$A\, e_{t+dt} = g_t \tag{7}$$

Note that contrary to what happened in the first model (equation (4)), the matrix A which appears hear is not a function of time. So it only needs to be factorized at $t = 0$. This saves substential computation.

Let us make some further comparisons between the two models[2]:

- We just underlined that the use of a linear formula to compute the energy is less time-consuming than a non-linear one. In addition, the equations for the hybrid model stay well conditioned when rigidity increases.
- Linear elasticity is less realistic than non-linear one, but proves to be sufficient for small deformations.
- This implies that the hybrid model is well adapted to animate deformable solids, but, contrary to the first model, cannot be applied to flexible surfaces such as clothes: the rigid reference component defines a rest-shape, and only small deformations from this shape are allowed.

2.3 Inelastic deformations

The equilibrium-shape of an object experiencing inelastic deformations is not only function of its initial shape and of the external forces which are applied. It also depends on all the history of deformations. In general, the object never recovers its original shape.

Terzopoulos and Fleischer [22] generalize the hybrid model of [25] in order to model some inelastic deformations. The reference component r becomes a function of time, and is slowly distorted.

- Viscoelasticity is obtained by letting:

$$\frac{dr}{dt}(u,t) = \frac{1}{\eta(u)}\, e(u,t) \tag{8}$$

  where $\eta(u)$ is the viscoelasticity constant at point u.
- During a plastic deformation, the reference component absorbs all the deformations which exceed a given limit.
- Fractures are modeled by introducing discontinuities when the local distortion exceeds a critic value. The points of possible fracture are preselected in the body description. A special process can be used for automatic propagation of fractures where the distortion is the most important.

Notice that these methods work by adding special processings for each inelastic deformation. So, the view of the different behaviors which is given is far from unified.

3 Layered models with discrete mechanical components

As a general rule, dynamic equations in continuous media don't seem well fitted to the animation of complex models imitating the real world, except perhaps if a very specific structure is designed[3]. Nevertheless, these complex objects, figures, animals, parts of the human body remain animation subjects of choice.

---

[2] Another discussion of these models can be found in [22].

[3] We will describe in Section 5 Gourret, Thalmann, and Magnénat-Thalmann's model, which simulates the contact between a human hand and a deformable ball.

Many authors choose approaches which still use some dynamic laws but are not based on any physical theory. Deformable objects are constructed by combining very simple mechanical elements (such as elementary masses linked by springs and dampers), with the possible addition of purely geometric components. These models are animated by integrating through time the differential equations of movement ($\Sigma\vec{f} = m.\vec{a}$) associated to each elementary mass of the system.

Sometimes specific to a fixed purpose, layered models are outstanding for the simplicity and efficiency of control they offer, and can yield quite convincing and natural-looking effects in interactive environments.

3.1 Luciani's standpoint: the system "CORDIS-ANIMA"

The "mass-damper-spring" formalism

The work of Luciani and Cadoz is derived from the following idea [17]: it is not always necessary to simulate the real physical structure of an object to reproduce its movement. The most important is to build a "mechanical representation" of the objects or phenomena to animate.

So, the authors describe their deformable objects as some elementary masses linked together by "conditional connections" (such as damped springs) whose parameters can change through time. The resulting system CORDIS-ANIMA [15, 17, 16] is both a modeling system and a simulation module for deformable objects.

Some advantages of this formalism are:

- An unified treatment of interactions inside an object and between the object and the external world (we will describe more precisely the interaction processing in Section 5),
- Real time animation, with enables direct manual intervention of the user. He manipulates an input/output device, which conveys the applied forces to the models, and feeds back their reactions.

Hybridization with geometric components

Some parts of a complex deformable object (a figure's skin for instance) play a insignificant part in the dynamic of movement. Luciani and Cadoz [17] propose to neglect the influence of these components, for which purely geometric models are sufficient.

In CORDIS-ANIMA, geometric points are modeled by zero-mass mechanical points. Three types of links are proposed to connect these components to the dynamic part of the objects (see figure 3)[4]:

- Rigid connections in a referential defined by three dynamic points,
- Geometric hinges: point P is computed from two other points by conservation of two distances,
- Plastic behavior: point P is computed from two other points in such a way that the surface of the triangle stays constant.

Geometric points are totally passive. Influenced by mechanic points, they can't react on them.

This idea of a geometric hybridization has been used again and generalized in a large proportion of the models described below. The main advantages of hybridization are:

[4]These connections were designed for a first version of CORDIS-ANIMA, which was 2-dimensional.

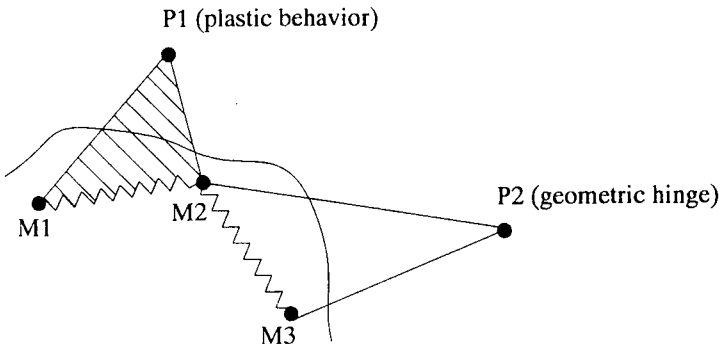


FIGURE 3. Geometric connections

- Optimization of movement control through suppression of useless degrees of freedom.
- Possible use of continuous representations for geometric components (parametric surfaces for instance).
- Better control of the animation. In particular, purely geometric deformations can be added to act on the geometric components[5].

3.2 Heating and melting deformable models

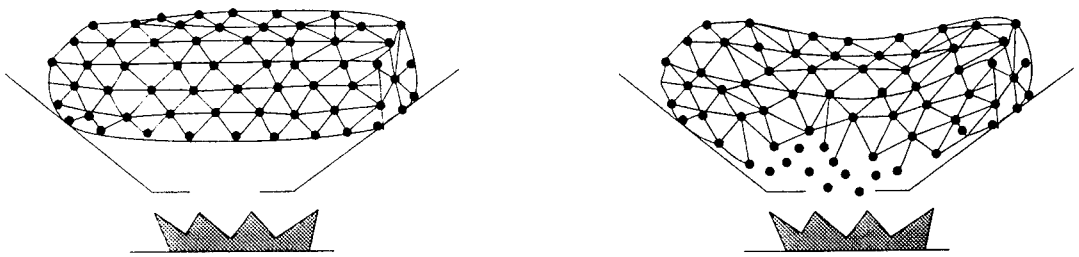


FIGURE 4. Melting a deformable model

In [24], Terzopoulos et al. use discrete elementary masses to model a deformable material which conducts heat, and has a fluid-like behavior in the molten state. The transition from solid to fluid is achieved by modifying the interactions between the masses:

- In the solid state, the material is a lattice of elementary masses linked by thermoelastic springs. These springs propagate the heat according to the "heat equation" (parabolic partial differential equation). Their stiffnesses are functions of the temperature of the connected masses. So, the material becomes less and less rigid while the heat increases. When their temperature reaches a fixed limit, the springs melt and disappear. See figure 4.
- As soon as an elementary mass is disconnected from its neighbours, its movement is computed by using molecular dynamic laws. Two types of interactions are used in the fluid state: long-range attraction forces and short-range repulsion forces.

[5]For instance, Chadwick, Haumann, and Parent use the "Free Form Deformations" of [21] in their model described in Section 3.4.

3.3 Animation of snakes and worms

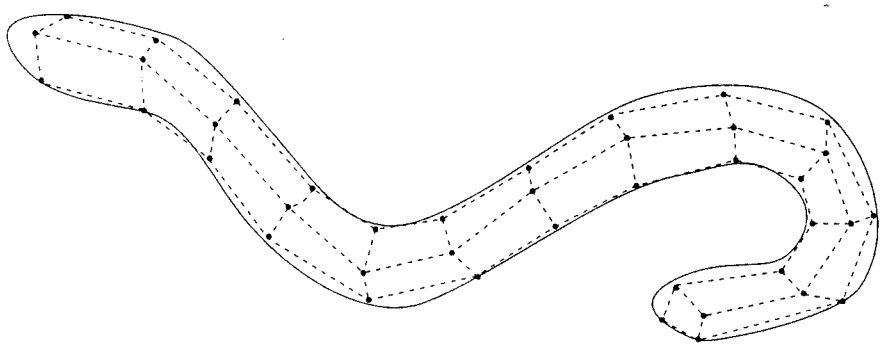


FIGURE 5. A deformable snake

Miller [18] uses an hybrid layered model with dynamic and geometric components to animate snakes and worms (see figure 5):

- The first layer is a mechanical module. The body of a snake is composed of a chain of paralellepipeds, whose vertices are elementary masses and whose edges and diagonals are damped-springs.
- The second layer is a purely geometric surface, which models the skin of the snake. It is generated by sweeping from the positions of the elementary masses.

The interpenetrations between the snakes and the rigid obstacles of the scene are avoided by moving the penetrating masses backward, while inverting their radial speeds.

To achieve realistic animations of creeping animals, Miller must find forces which can be applied on the springs to simulate the action of muscles. He chooses sinusoïdal contraction forces. Directional friction forces are added to model the scales which prevent the snake from moving backward.

3.4 Modeling muscles and fatty tissues for figure animation

In 1976, Burtnyk and Wein [6] presented a method for putting a geometric skin on a skeleton animated with key-frames. Chadwick et al. [8, 7] implement a model of that kind for figure animation. Dynamic layers are integrated to simulate various types of muscles and fatty tissues.

The model is constituted of three interrelated layers:

- The first layer is an articulated rigid skeleton, whose movement is precomputed. It may be a dynamic skeleton governed by rigid bodies physical laws or a geometric one animated with key positions.
- The second layer models the deformable flesh of the figure. Geometric deformations are used to model muscles, while dynamic ones are applied to fatty tissues:
  - Each muscle is modeled with two Free Form Deformations (or FFDs) defined in [21]. The displacement of the FFD lattice's control vertices are computed from the angle at the articulation join associated with the muscle. Notice that it is not the muscles which govern the movement of the skeleton, but the contrary !

- Visco-elastic deformations of fatty tissues are obtained by connecting together the control vertices of a FFD with damped springs. The properties of FFD lattices insure that these dynamic deformations are automatically translated into deformations of the tissues they contain.

The user can apply lower level control by moving manually the control vertices of the FFDs.

- The skin of the figure is represented by a purely geometric surface, deduced from the position of the second layer.

The authors emphasize on the interest of partially dynamic and partially geometric models in order to animate very complex objets. The use of a layered structure provides simple and efficient control of movement and deformations.

3.5 Animation of "articulated deformable objects"

Gascuel, Verroust and Puech [9, 27, 11] propose a somewhat general model for complex deformable objects whose elastic flesh coats an articulated skeleton.

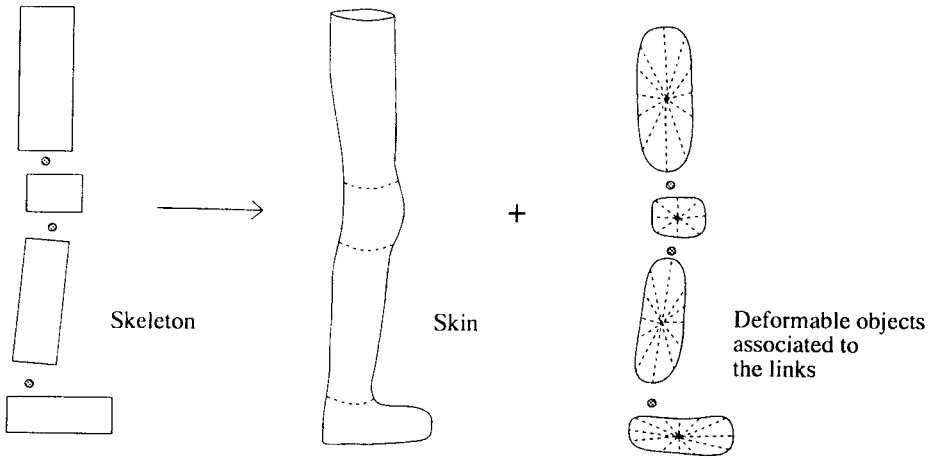


FIGURE 6. Automatic coating of a skeleton

The model is structured into three dynamic and geometric layers (see figure 6):

- The first layer is a rigid articulated skeleton composed of links connected by hinges. The movement of this skeleton is not precomputed as in Chadwick et al.'s method, but generated from dynamic laws during the animation of the articulated deformable object.

- Deformable components are associated to each link of the skeleton, in order to model deformable flesh. Each deformable component is structured into two modules:

  - The "basic mechanical module" simulates the axial deformations of the flesh. It is designed by cones of deformable flesh arranged in a star-shape way around the center of mass of the link. The deformations of these cones are governed by damped-springs.
  - A second layer models the propagation of deformations from one flesh cone to the others. This can be achieved by using dynamic criteria as well as geometric ones (such as constant volume, or constant surface area deformations).

These deformable components are particularly convenient to process automatic detection and response to collision (we will come back to that point in Section 5).

- A purely geometric skin represented by a Bspline surface covers the deformable elements. The control vertices of this surface are located at the extremities of the axial springs. In consequence, deformations of flesh components immediately translate into skin deformations.

An "automatic coating method" is provided [11] to construct very easily articulated deformable objects from a skeleton described by the user.

The animation sequence is computed by applying a simultaneous dynamic simulation of the interrelated levels of the model. In consequence, the movement of skeleton can be modified after a collision (by applying the response forces computed at the flesh level).

4 Controlling flexible models with behavior constraints

When we deal with dynamic animation, one of the main difficulties is to obtain a convenient and efficient control of movement and deformations:

- If the objects deform according to continuous elasticity laws, it is rather impossible to specify any global constraint on the material which is simulated (such as constant volume deformations for instance). The same problem occurs for most of the layered models.
- It is often useful to move a deformable object along a predefined path. Finding which forces and torques must be applied to do so is not easy.

Let us look at some constraint methods that have been proposed to cope with these problems.

4.1 Penalty methods

Total energy of moving bodies is a decreasing function through time (for all dissipative models). In consequence, dynamic animation can be viewed as an energy minimization problem. Penalty methods consist in adding "constraint terms" to this energy, in order to penalize the violations of the constraints.

Such methods have been applied in the field of dynamic animation of articulated rigid objects [5, 29], and also for object reconstruction [26]. They can be used as well to constrain deformable models.

In [20], Platt and Barr list the advantages they offer:

- Several constraints can be applied at the same time;
- Penalty method compromise between constraints, even if they are not compatible;
- No extra differential equation is required.

They also emphasize on several drawbacks:

- The constraints are never completely reached,
- The equations become ill-conditioned when constraint strengths are increased (through the associated coefficients in the energy formula).

We will study now two other constraint methods that were introduced by Platt and Barr [20] to cope with these limitations.

4.2 Reaction constraints

This is a compromise between projection methods and inverse dynamics. At each time step, the system computes the force which must be added to each finite element or each elementary mass of the model in order to fulfill the constraint. Let $D = \vec{M_0M}$ be the vector representing deviation between the next position of the element (eventually violating the constraint) and the position it should have. After suppressing the component of the applied forces which is parallel to D, we add the exact "constraint force" which will force the element to fulfill the constraint at the next time step.

This technique is different from the method used by Barzel and Barr [3, 2, 4] in the case of articulated rigid bodies. Here, the added force is not aimed at slowly minimizing the distance to the constraint, but prevents constraint violations in a single time step.

The reaction constraints enable to enforce mass elements to follow a predefined path, parametrized by time (then, the object will move as if it was dragged by these points). They can also been used to avoid the interpenetrations between a deformable body and a rigid polyhedron. This application will be discussed in Section 5.

Let us list some advantages of reaction constraints:

- These constraint are immediately and perfectly verified,
- Adding constraint forces is not time consuming (for the very simple constraints given behind, these forces are easy to compute).

Nevertheless, the method shows some limitations:

- You can't apply several constraints at the same time to the same mass element,
- This technique does not enable to simulate high-level properties of the deformable material (such as uncompressibility).

4.3 Augmented lagrangian constraints

Lagrange multipliers (noted λ) can be introduced to transform a constrained optimization problem into a simple one. To do so, extra differential equations are added.

More precisely, let $x = (x_1, x_2, ..., x_n)$, and $f(x)$, $g(x)$ be two real functions. The problem:

$$\text{minimize } f(x) \text{ with the constraint } g(x) = 0 \tag{9}$$

leads to the following Lagrange equations:

$$\dot{x}_i = -\frac{\partial f}{\partial x_i}(x) - \lambda \frac{\partial g}{\partial x_i}(x) - cg(x)\frac{\partial g}{\partial x_i}(x) \tag{10}$$

$$\dot{\lambda} = g(x) \tag{11}$$

These equations can be generalized to the cases of multiple constraints and inequality constraints (see [20] for more details).

The augmented lagrangian technique proves to be useful to specify global properties of the deformable material:

- Constant volume deformations are designed by making each finite element uncompressible,
- Plastic materials are modeled by using inequality constraints.

In conclusion, the augmented lagrangian method is time consuming (because of the addition of new differential equations and new unknowns), but is farther more general than the reaction constraint method:

- Several constraints can be applied at the same time,
- These constraints can be non-linear, or given by inequalities.

5 Collision detection and response

Automatic treatment of collisions is an important topic in order to understand the real usefulness of the deformable models for animation purposes.

Most animation systems do not offer automatic detection of collisions. After having specified the trajectories of the objects, the user may see, to his/her surprise, two of them passing quietly through each other.

In addition to the purely kinematic problem of interaction detection, dynamic models are facing a more fascinating challenge: they must respond automatically and in a realistic way to possible collisions. A dynamic system which does not include these abilities, such as Chadwick et al.'s model [8, 7], is therefore limited to the animation of a single isolated object. In some systems, tricks are used to prevent the objects from penetrating rigid still obstacles. For instance, Terzopoulos suggests in [23] that they be surrounded by exponential force fields. Miller [18] avoids the penetrations of the snakes under the floor by displacing some elementary masses and inverting their radial speeds. He emphasizes that this solution would not work for stair-shaped obstacles, or for contacts between several snakes: the detection would become too inaccurate.

Finding a good response to collisions is especially difficult for deformable models. Contrary to the rigid case, collisions are not instantaneous nor conservative, a portion of the energy being consumed in deformations. Furthermore, it is not sufficient to compute the new speeds and accelerations of the objects: we also need their new shapes during and after the contact. The physical elasticity theory only describes the behavior of deformable material during small oscillations around equilibrium states. No real answer is given to the collisions problem. Moreover, up to now, the amount of computation needed deterred anyone from solving the systems of coupled elasticity equations for all the objects involved in a collision. So, even for models based on the elasticity theory, treatment of interactions can be applied only after spatial discretization. This increases collision detection problems as the objects are represented by networks of sample points.

Nevertheless, several interesting solutions have been introduced during the past few years:

- In 1988, an attempt is made by Platt and Barr to use a constraint method to simulate interactions. Nevertheless, this technique is restricted to the collision between a deformable model and a rigid polyhedron.
- The same year, Moore and Wilhelms present a general solution, based on artificial springs introduced during collisions.
- In 1989, Gourret, Thalmann, and Magnénat-Thalmann simulate a hand in a grasping task with a finite element model.
- Luciani proposes two ways of optimizing interaction detection and response for discrete mechanical systems (1988,1989).

- A new method is associated to the articulated deformable objects of Gascuel, Verroust and Puech (1990).

The remainder of this Section describes these models and discusses their advantages and limitations.

5.1 Simulating interactions with constraint methods

In [20], Platt and Barr propose the use of constraints to compute the interactions between a deformable model and a rigid polyhedron, mobile or not. The deformable model can be built with a continuous deformable material (as in [23]), or with discrete mechanical components. The method uses the "reaction constraints" described in Section 4. When a mass element of the deformable model attempts to penetrate the rigid polyhedron, its externally applied forces are modified. The added force moves the mass to the surface of the polyhedron, and so the constraint stays fulfilled (this method doesn't work with non-polyhedral rigid obstacles, because computing the associated constraint forces would become too difficult). An opposite force is applied to the rigid polyhedron (if it is a dynamic model), in accordance to the action and reaction principle.

The reaction constraint method is not based on any physical law: The force which is arbitraryly added does not look like a "reaction force" (in spite of the name of the method): it appears before contact; it is not computed from the parameters of a collision (such as speed, kinetic energy just before the impact, local stiffness of the deformable object...), but depends on all the external forces that are already applied on the elementary mass.

This technique has been employed by Terzopoulos et al. [24] to avoid that the particles of a melting material pass through the rigid container in which it is heated (this application involves contacts rather than collisions). The results are quite natural-looking.

Nevertheless, the reaction constraint technique is not very general. Only a single constraint can be applied at the same time to each elementary mass. Moreover, this method does not seem easy to generalize to a collision between two deformable objects, or with a non-polyhedral solid, the constraints becoming too complex.

5.2 Moore and Wilhelms' methods for collision detection and response

Moore et Wilhelms describe in [19] a set of methods for interaction detection and response. Some of them can be applied to deformable models based on the elasticity theory as well as to discrete models.

Collision detection

Suppose that the deformable bodies are described by a lattice (whose vertices can be elementary masses of finite elements' nodes). Their external surfaces are then defined by a set of triangles constructed on some of these vertices.

The detection algorithm (see figure 7) consists in testing if the trajectory of each point of an object during the last time step did not pass through a triangle not containing this point (so, auto-intersections are detected as well). If n is the number of triangles and m the number of points, the basic detection is in $O(nm)$.

A choice can be made according to the accuracy of detection which is needed:

- If we consider than the triangles are fixed during the time step, the method consists in testing intersections between triangles and segments (which represent the trajectories of points during the time step). This is done by solving a system of three equations and three unknowns, u, v (barycentric coordinates defined by the

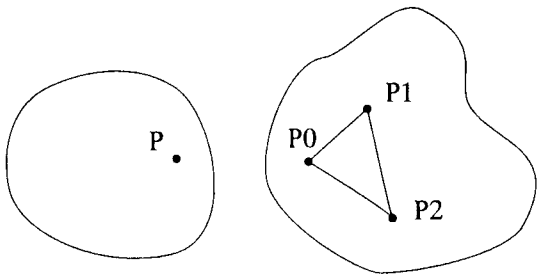


FIGURE 7. Collision detection

triangle) and t (date of the collision) :

$$P + (P' - P)t = P_0 + (P_1 - P_0)u + (P_2 - P_0)v \tag{12}$$

Then, the following conditions must be verified:

$$\begin{cases} 0 \leq t \leq 1 \\ u \geq 0 \\ v \geq \\ u + v \leq 1 \end{cases} \tag{13}$$

However, in most cases, the simplification stipulating that the triangles don't move while the points move is not justified.

- If the triangles move, equation (12) becomes:

$$P + Vt = P_0 + V_0 t + ((P_1 - P_0) + (V_1 - V_0)t)u + ((P_2 - P_0) + (V_2 - V_0)t)v \tag{14}$$

where V, V_0, V_1, and V_2 are the speeds of the vertices.

By eliminating u and v in (14), we obtain an equation of degree 5 in t. The actual time of the impact is computed by binary search; then, u and v are evaluated.

Several solutions are proposed to improve these methods: the use of bounding boxes/-spheres, hierarchical representations with octrees, etc.

Anyhow, it is impossible to escape the inefficiency inherent in a point-wise detection.

When a collision is detected, an interpenetration has already occurred between the two objects. Moore and Wilhelms do not say if we must go back in time to the date of the impact before continuing the simulation.

Collision response

From the techniques proposed by Moore and Wilhelms to compute the response to collisions, only one can be applied to deformable objects. It consists in putting a temporary spring between the closer points of the two objects. The contraction of this spring will simulate a kind of response force (see figure 8).

The authors do not precise when this spring must be introduced (before or after the interpenetration), when it must be removed, nor how its stiffness k must be chosen. Nevertheless, they say that if deformable objects are involved, the value k_r of the stiffness after the impact must be smaller than the stiffness k_a used before:

$$k_r = \epsilon k_a \tag{15}$$

where ϵ (chosen between 0 and 1) is a function (to be specified) of the stiffness of the material.

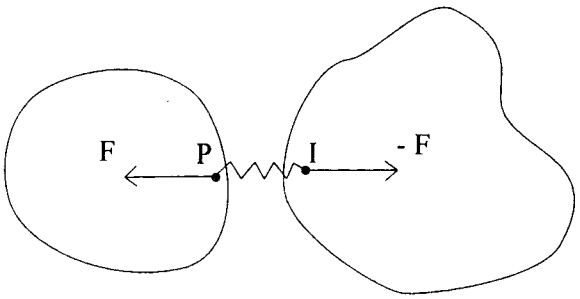


FIGURE 8. Collision response

According to [28] this model of collisions is time consuming especially if stiff springs are used (time steps must be very small to avoid divergences). In addition, if we want to use this solution to model a contact which lasts in time, the use of a spring could produce unwanted oscillations.

5.3 Simulation of a human hand in a grasping task

The system developed by Gourret, Thalmann, and Magnénat-Thalmann [12] [13] is aimed at simulating the equilibrium deformations due to the contact between a hand and a deformable ball. It is based on the elasticity theory and finite element methods.

First of all, notice that this is not a classical dynamic simulation. The displacements of the hand are not computed from externally applied forces. At the contrary, they are governed by the skeleton's key positions specified by the user.

Description of the model

The simulation is based on the principle of virtual displacements. Let R represent the internal forces, and let V, S and F respectively be the volume, surface, and pinpoint external forces. Then, the virtual works of these forces obey the equation:

$$wR = wV + wS + wF \tag{16}$$

Equation (16) is discretized by splitting the hand flesh and the deformable ball into finite elements. Some of the hand finite elements are attached to its rigid skeleton (designed with a very realistic shape). At equilibrium, or during small oscillations, the relation between deformations and the internal elasticity force is:

$$KU = R \tag{17}$$

where K is the stiffness matrix (its size is proportional to the number of finite elements), and U is the displacement vector from the unloaded configuration.

Simulation module

The computation is based on the following idea: when two objects are in contact, they can be viewed as a single object. Thus, equation (17) can be used to simulate the global behavior of the set on finite elements representing the hand and the ball.

The algorithm used can be decomposed as follows:

1. Displace the hand's flesh according to a position of the skeleton specified by the user (without considering interpenetrations with the ball).

2. Start a series of iterations:

(a) Deform the ball from its rest shape to avoid interpenetrations. This is achieved by displacing each reference vertex of the ball which is located inside the hand.

(b) Compute the displacements of the finite elements inside the ball, using equation (17). Compute the reaction forces applied by the ball onto the hand.

(c) These forces being fixed, solve globally equation (17), for the set of finite elements corresponding to both objects. Compute the reaction forces that are applied onto the bones. Displace the flesh of the hand.

Stop :

- If the forces applied onto the bones are too strong to be physically realistic (then, the user must change the position he specified),
- If the finite elements are close enough to an equilibrium state (eg. if the displacement computed during the last iteration is very small).

Discussion

Let us emphasize some of the advantages of this method:

- The control technique is clever: the user can more easily describe a precise movement of the hand by specifying skeleton key-positions rather than forces to apply. Moreover, the system detects unrealistic movements.
- The user can cut from all the parameters that make the movement very complex, eg. the deformable nature of the ball and of the hand flesh. In addition, he is sure that the animation will be much more realistic than if he had designed it himself.
- All in all, a very natural-looking and impressive animation of a complex unhomogeneous object (a hand) which interacts with the external world is obtained.

Nevertheless, this method does not seem suitable for a current use in animation systems:

- A finite element model is very realistic, but also very complex. It must be reconstructed for each particular scene. This also includes the search for suitable limit conditions.
- For each frame, the amount of computations before reaching an equilibrium state seems quite important. In particular, each iteration (steps (a) to (c)) includes a point-wise detection (and displacement) of the ball's node that have penetrated inside the hand. The authors do not give the approximative number of iterations before converging. Anyhow, we can believe that the animation is not computed in real time, particularly because of the number of finite elements used.
- Finally, and above all, the method is restricted to a soft and lasting contact between two objects. In particular, it would not work to model a collision, which cannot be simulated through a succession of equilibrium states. In consequence, an animation system using the previous technique would have to integrate another method especially devoted to collisions, and an artificial limit would have to be fixed between "true collisions" and contacts. This problem, which also occurs for rigid objects [1], can be especially tricky in the limiting cases.

5.4 Interactions in discrete systems: Luciani's approach

We saw in Section 3.1 that the objects used in CORDIS ANIMA are based on conditional connections linking elementary masses. In such a model, interactions between objects can be viewed as particular kinds of connections. Indeed, each couple of mass points belonging to different objects is always linked by a spring-like connection. The stiffness of this spring is zero when the masses are far enough from each other. It increases when the masses gather, in order to simulate the reactive force due to a collision.

"Transmitter elements"

When dealing with discrete objects, an accurate detection of interactions can be difficult to achieve, particularly when geometric components are added:

- Geometric points are often used to model the "skin" of an object, but as mentioned previously, are purely passive. They cannot be used to detect interactions, because they cannot influence the movement of the mechanical components.
- Moreover, the mechanical components are themselves unadapted to interaction detection: they can be rather far apart, and very distant from the external shape of the object.

This problem rises in all layered models based on discrete mechanical elements. For instance, Miller in the snakes animation system cannot use the geometric skin to detect contacts, by lack of feedback from the skin to the mechanical components; a detection with the elementary masses, which is not very accurate, has to be used instead (see [18]).

To cope with this problem, Luciani [16] introduces a third kind of components, the "transmitter elements", which are sorts of "access points" through which the objects interact. They can be chosen within the mechanical points, or simply dynamically connected with these points (because a collision must modify the movement of the mechanical part of the objects). An interaction will be detected if two transmitter points of different objects become close enough. So, an accurate detection near the geometric components can be obtained by associating transmitter elements with them.

The use of a third type of point complicates the parameters left to the user during the modeling phase, but increases the richness and versatility of the models. Moreover, it is a way of minimizing the number of interaction tests, by limiting them to transmitter points located on the surface of the objects.

"Agglomerates" of elementary particles

Jimenez and Luciani [14] proposed in 1989 another method for optimizing interaction detection. The objects are modeled by agglomerates of elementary spheric particles (see figure 9).

Each agglomerate includes three successive layers of deformable material, characterized by the interaction laws between their particles:

- The "kernel", formed by a few large particles, represents the more rigid internal part of the object.
- The "derm" imitates the deformable flesh. It is modeled by particles attracted by the kernel, but two by two repulsive. These interaction laws produce an uniform distribution of the derm around the kernel.

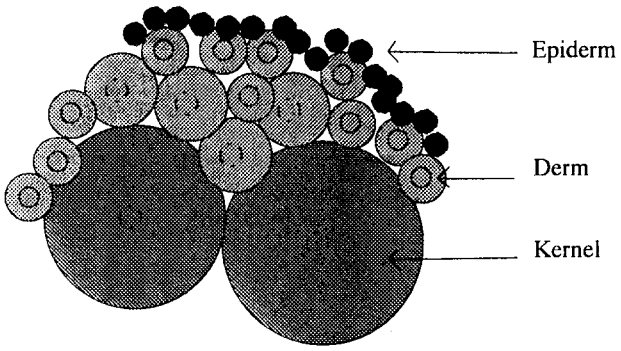


FIGURE 9. Agglomerate

- The "epiderm" insures the cohesion of the agglomerate. The attractive interaction law which is used simulates the tension of the skin. All the interactions between the agglomerate and the external world are treated at the epiderm level (except if we want to enable fractures).

Let us list the main advantages of this model:

- Numerous physical phenomena - such as flexibility, plasticity, collisions, and fractures - can be modeled with agglomerates. The different behaviors are obtained by playing with the interaction laws between the different kinds of particles.
- Based on point-wise physics, this model does not introduce any complex computation (no rotation, no vector product, no projection...). Moreover, the equations of movement can be solved in parallel for each particle (once the applied forces due to interactions are known). This allows the real time animation of complex agglomerates.

5.5 Interactions between articulated deformable objects

The structure of articulated deformable objects proposed by Gascuel, Verroust and Puech (and described in Section 3.5) is particularly well adapted to interaction detection and response. As mentioned, that each link of the skeleton is coated with a deformable component, constituted of deformable cones (which are modeled by springs and arranged in a star-shape way). Interactions are detected and treated independently for each of these components. The resulting reaction forces are transmitted to the associated links, in order to adequately modify the movement of the skeleton.

Let us describe the algorithm more precisely:

1. Detect interactions:
   - Predetection with bounding boxes/spheres,
   - Test if the extremities of the springs associated to the deformable components are inside another object, deformable or not.
2. Avoid interpenetrations by computing the new shapes of the deformed objects (use their local relative stiffness on the contact zone).
3. Respond to collisions:
   Compute the reaction forces that must be applied to the skeleton's links according to the deformations of the flesh cones touching another object (this computation is especially simple, because the cones are modeled by springs).

Of particular interest are the following remarks:

- This method offers a simple and efficient control of the angle constraints imposed at articulation points: if an angle becomes too small, a collision will be detected between two neighbouring deformable elements, and the response forces will prevent the angle from becoming smaller (see [10, 9] to find a technique for precise angle constraint control).

- A succession of small collisions is not a good model for contacts: equilibrium positions can be hard to reach. For instance, unwanted oscillations can appear when an object is resting on the floor. So, as the contact lasts, we introduce reaction forces computed with the associated rigid skeleton. The progressive introduction of these forces can lead easily to an equilibrium state, while the deformable component flattens on the floor (see [27] for more details).

6 Conclusion

Deformable models based on pure simulation of dynamic equations offer unsurpassed realism. Nevertheless, their use is not evident for a layman:

- The physical parameters are often non-intuitive. A good knowledge in physics is needed to understand them (for instance, the effects of the Lamé parameters on an elastically deformable body are not obvious).

- They do not fit exactly with the notions we would want to control. In particular, specifying global properties of a deformable material —such as uncompressibility— is not easy.

In fact, these models seem unadapted to the simulation of heterogeneous complex objects from the real world, except perhaps if a specific structure is built for each animation sequence (as was done by Gourret et al. for a hand in a grasping task). However, this solution would be computer intensive and time consuming.

On the other hand, modular models combining discrete mechanical components and geometric layers are less "realistic", but simplify the modeling and the animation of complex objects. The fact that the parameters from different layers can be controlled independently can be very convenient for the precise tuning of the structure. Discrete models are not suitable for the simulation of existing materials. However, this limitation is also their richness: they can imitate behaviors for which no adequate answer is given by physics (it would be difficult to imitate a creeping snake, muscles, or a collision between two articulated deformable objects with mathematicals physics!). Furthermore, very simple dynamic equations enable the interactive computation of animation sequences. So, the user can modify immediately his parameters according to the desired effect, realistic or not.

This state of the art emphasized on two highly challenging problems:

- The control of movements and deformations for active models:
  It can be improved through optimization techniques, but these methods are often computer intensive. In addition, the constraints must be pre-programmed before being used, and this can be quite difficult for a non-specialized user.

- The automatic detection and response to interactions:
  Some interesting solutions were proposed to deal with interactions between several

deformable models. However, they are not always generaly applicable. Besides, some of them are based on the introduction of non-intuitive mechanical elements during periods of contact.

In the next few years, these two problems will probably remain important scopes of research, especially through the apparition of new hybrid models where geometric specifications will collaborate with dynamic components.

7 References

[1] D. Baraff. Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies. *Computer Graphics*, 23(3):223–232, July 1989.

[2] A. Barr, B. Von Herzen, R. Barzel, and S. Snyder. Computational Techniques for the Self Assembly of Large Space Structures. In *8th Princeton/SSI Conference on Space Manufacturing*, Princeton New Jersey, May 1987.

[3] R. Barzel and A. Barr. Modeling with Dynamic Constraints. *State of the Art in Image Synthesis (SIGGRAPH'87 course notes Number 17, Anaheim, Ca)*, 1987.

[4] R. Barzel and A. Barr. A Modeling System Based on Dynamic Constraints. *Computer Graphics*, 22(4):179–188, August 1988.

[5] L. Shapiro Brotman and A. N. Netravali. Motion Interpolation by Optimal Control. *Computer Graphics*, 22(4):309–315, August 1988.

[6] N. Burtnyk and M. Wein. Interactive Skeleton Technique For Enhancing Motion Dynamics in Key Frame Animation. *Communications of the ACM*, 19(10):564–569, October 1976.

[7] J.E. Chadwick, D.R. Haumann, and R.E. Parent. Layered Construction for Deformable Animated Characters. *Computer Graphics*, 23(3):243–252, July 1989.

[8] J.E. Chadwick and E. Parent. Critter Construction: Developping Characters for Computer Animation. In *PIXIM 88*, pages 283–305, Paris, France, October 1988.

[9] Marie-Paule Gascuel. Déformations de surfaces complexes : techniques de haut niveau pour la modélisation et l'animation. *Thèse de doctorat*, Université Paris XI, October 1990.

[10] M.P. Gascuel, A. Verroust, and C. Puech. Animation with collisions of deformable articulated bodies. In *Eurographics Workshop on Animation and Simulation*, September 1990.

[11] M.P. Gascuel, A. Verroust, and C. Puech. A modeling system for complex deformable bodies suited to animation and collision processing. *Journal of Visualization and Computer Animation*, 2(3), August 1991.

[12] J.P. Gourret, N. Magnenat Thalmann, and D. Thalmann. Simulation of Object and Human Skin Deformations in a Grasping Task. *Computer Graphics*, 23(3):21–29, July 1989.

[13] J.P. Gourret, N. Magnenat Thalmann, and D. Thalmann. The Use of Finite Element Theory for Simulating Object and Human Body Deformations and Contacts. In *Eurographics 89*, pages 477–487, September 1989.

[14] S. Jimenez and A. Luciani. Une vue unifiée des comportements de multitude, flexibilité, plasticité et ruptures : Billes, bulles et agglomérats. *Rapport de Recherche ACROE*, 89(15), November 1989.

[15] A. Luciani. Un Outil Informatique de Création d'Images Animées. *Thèse de docteur ingénieur d'électronique*, Institut National Polytechnique de Grenoble, November 1985.

[16] A. Luciani. Modèles pour la synthèse d'images animées. *Rapport de Recherche ACROE*, 88(5), January 1988.

[17] A. Luciani and C. Cadoz. Utilisation de Modèles Mécaniques et Géométriques pour la Synthèse et le Contrôle d'Images Animées. In *Deuxième Colloque Image, CESTA*, Nice, April 1986.

[18] Gavin S.P. Miller. The Motion Dynamics of Snakes and Worms. *Computer Graphics*, 22(4):169–177, August 1988.

[19] M. Moore and J. Wilhelms. Collision Detection and Response for Computer Animation. *Computer Graphics*, 22(4):289–298, August 1988.

[20] J.C. Platt and A.H. Barr. Constraint Methods for Flexible Models. *Computer Graphics*, 22(4):279–288, August 1988.

[21] T.W. Sedeberg and S.R. Parry. Free-form Deformations of Solid Geometric Models. *Computer Graphics*, 20(4):151–160, 1986.

[22] D. Terzopoulos and K. Fleischer. Modeling Inelastic Deformations: Viscoelasticity, Plasticity, Fracture. *Computer Graphics*, 22(4):269–278, August 1988.

[23] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically Deformable Models. *Computer Graphics*, 21(4):205–214, July 1987.

[24] D. Terzopoulos, J. Platt, and K. Fleisher. Heating and Melting Deformable Models (From Goop to Glop). In *Graphics Interface'89*, pages 219–226, London, Ontario, Canada, June 1989.

[25] D. Terzopoulos and A. Witkin. Physically Based Model with Rigid and Deformable Components. *IEEE Computer Graphics and Applications*, pages 41–51, December 1988.

[26] D. Terzopoulos, A. Witkin, and M. Kass. Energy Constraint on Deformable Models. *State of the Art in Image Synthesis (Siggraph'87 course notes Number 17)*, July 1987.

[27] Anne Verroust. Etude de problèmes liés à la définition, la visualisation et l'animation d'objets complexes en informatique graphique. *Thèse d'état*, Université Paris XI, December 1990.

[28] J. Wilhelms, M. Moore, and R. Skinner. Dynamic Animation : Interaction and Control. *The Visual Computer*, 2(4):283–295, December 1988.

[29] A. Witkin and M. Kass. Spacetime Constraints. *Computer Graphics*, 22(4):159–168, August 1988.

Chapitre 2

Displacement constraints for animating articulated structures

Ce travail sur maintien de contraintes géométriques entre objets au cours d'une simulation a fait l'objet de deux publications, la première en version courte dans un colloque, la seconde, plus complète, dans une revue internationale à comité de lecture :

[GG92] Displacement Constraints: A New Method for Interactive Dynamic Animation of Articulated Solids. Jean-Dominique Gascuel and Marie-Paule Gascuel. Dans les actes de *Third Eurographics Workshop on Animation and Simulation*, Cambridge, Royaume-Uni, septembre 1992.

[GG94] Displacement Constraints for Interactive Modeling and Animation of Articulated Structures. Jean-Dominique Gascuel and Marie-Paule Gascuel. Dans *The Visual Computer*, volume 10-4, mars 1994.

Résumé

Cet article présente un ensemble cohérent de méthodes pour l'assemblage automatique et l'animation interactive de systèmes de solides vérifiant des contraintes géométriques. Ces méthodes permettent de modéliser des structures articulées dont chaque connexion possède un nombre quelconque de degrés de liberté en rotation et en translation, chacun d'eux pouvant être muni de butées. L'animation est calculée en découplant le mouvement libre de chaque solide de l'action des contraintes, qui sont maintenues grâce à des ajustements itératifs en déplacements. Actuellement implémentée dans un système d'animation dynamique, la méthode prend en compte les paramètres physiques des objets dans la phase de traitement des contraintes. En particulier, les moments du premier ordre du système sont conservés. L'approche pourrait être facilement étendue à un système de modélisation ou d'animation n'utilisant pas de modèles physiques d'objets, en laissant simplement plus de paramètres sous le contrôle de l'utilisateur.

Displacement constraints for interactive modeling and animation of articulated structures

Jean-Dominique Gascuel* and Marie-Paule Gascuel*

LIENS, CNRS URA 1327,
Ecole Normale Supérieure, 45 Rue d'Ulm,
F-75230 Paris Cedex 05, France

This paper presents an integrated set of methods for the automatic construction and interactive animation of solid systems that satisfy specified geometric constraints. Displacement contraints enable the user to design articulated bodies with various degrees of freedom in rotation or in translation at hinges and to restrict the scope of the movement at will. The graph of constrained objects may contain closed loops. The animation is achieved by decoupling the free motion of each solid component from the action of the constraints. We do this with iterative tunings in displacements. The method is currently implemented in a dynamically based animation system and takes the physical parameters into account while reestablishing the constraints. In particular, first-order momenta are preserved during this process. The approach would be easy to extend to modeling systems or animation modules without a physical model just by allowing the user to control more parameters.

Key words: Constraints – Modeling – Animation – Simulation – Dynamics

* *Present address*: iMAGIS, IMAG, BP 53, F-38041-Grenoble Cedex 09, France
Correspondence to: J.-D. Gascuel

The Visual Computer (1994) 10:191–204

1 Introduction

Animation systems in computer graphics must provide both modeling tools to define objects and methods to generate their movements and deformations. Purely descriptive animation systems are not always sufficient to animate complex structures (e.g. solids connected by various kind of joints). Various "active models" have been proposed to cope with this problem. Some are based on a set of simplified mechanical laws and most offer an unsurpassed realism. However, there is still no universal method that adapts to the great variety of objects and of motions and that combines efficiency with simplicity of use.

This paper presents a new way of maintaining geometric constraints between solids that can be used for both automatic assembly and interactive animation of articulated structures. At each time step of an animation, the solid components of the structures are first animated as if they were independent. Then the constraints are reestablished by a series of small corrections in displacement. Correction of the displacement, rather than evaluation of the contraint forces, leads to an algorithm which can be applied either to physically based animation systems (the corrections will take physical models into account) or to systems without physical models. The displacement-constraint approach is very easy to implement, provides a simple way of defining complex structures, and produces animations at interactive rates.

1.1 Related work

The animation of articulated solids has drawn a lot of attention in the past few years. [A detailed state-of-the-art can be found in Verroust (1990)]. Featherstone (1983) introduced an animation technique for articulated chains with components that possess one degree of freedom with respect to their parent. These chains are animated by forces and torques chosen by the user. Lathrop (1986) extended this method to tree-like structures and to graphs of components.

Wilhelms and Barsky (1985) proposed an approach adapted to tree-like structures that enables rotations and translations at each joint.

Armstrong and Green (1985) present a more efficient method that is restricted to rotations at hinges. Like the former approach, this does not work if the structure contains closed loops.

Using d'Alembert's virtual work principle, Isaac and Cohen (1987, 1988) animated articulated bod-

ies the components of which can have six degrees of freedom with respect to their parent. Motion is controlled by specifying some of the accelerations (inverse dynamics) and some of the external forces.

Barzel and Barr (1988) use geometric constraints connecting independent solid components to construct complex structures. The basic idea consists in evaluating the forces caused by the action of the constraints. To do this, a system of coupled differential equations (each of them corresponding to a constraint) is numerically integrated over time. The animation sequences that are shown illustrate the use of this method for automatic assembly processes: the successive configurations of the solids obtained during the computations are displayed, and the constraints are satisfied only at the last time step. To produce a full animation of articulated structures with this approach, the numerical integration process should be applied for each frame. The authors admit that their approach, although more general, turns out to be less efficient than the methods of Armstrong and Green (1985) or of Isaac and Cohen (1987, 1988).

More recently, van Overveld (1990) introduces a simple and efficient approach, perhaps less realistic than the previous ones because it uses point dynamics only. Each object is composed of a set of points with masses, connected by distance or angle constraints (so that the mass of an articulated object is divided among the hinges). The basic idea for the animation consists in decoupling the external actions on the system from the internal actions due to constraints. Van Overveld (1991), approximates solids with families of such points, linked by constraints that maintain rigidity. Points shared by two of these solids provide hinges with three rotational degrees of freedom.

1.2 Overview

The displacement-constraints method presented in this paper attempts to find a new compromise between the efficiency and the "realism" of an animation involving articulated structures. In order to model a large variety of objects and of motions easily, we animate solid components connected by geometric constraints. The user is free to choose the number of degrees of freedom in rotation and in translation at hinges and to specify angular or linear constraints on the motion. The graph of constrained objects may contain closed loops. A technique for the automatic construction of valid initial positions is provided with the animation method.

Our basic idea is to decouple the free motion of the solid components from the action of the constraints that are met through *iterative tunings in displacements*. These displacements include small translations and small rotations. To preserve the correctness of motion in an animation system based on solid mechanics, the corrections reestablishing constraints must take the physical parameters of the solids into account. In particular, they must maintain the system first-order momenta.

The correction of displacements rather than the computation of constraint forces is the central innovative aspect of the method. Thanks to this idea, the displacement-constraint approach is not restricted to physically based systems. No integration of constraint forces is required, and the few physical parameters used for constraint processing can be replaced easily by intuitive user's parameters. The method would work in a modeling system, or in an animation module using a non-physical model (for instance, one based on inverse kinematics or on behavioral approaches). In a comprehensive animation system, displacement contraints would be particularly useful to connect physical and non-physical objects.

The remainder of this paper develops as follows: Section 2 presents the basic ideas of our approach and motivates our choices. We give general algorithms for both animation and automatic search of valid initial positions. Section 3 deals with "point to point" constraints between solids that are used to create hinges with three rotational degrees of freedom. Section 4 extends the method to constraints that enable translations at hinges, possibly limited in scope by the user and shows how rotations can be restricted to fixed angular domains. Section 5 gives the details of the implementation and the user's interface and it comments on some results. Section 6 concludes and discusses the work in progress.

2 The displacement-constraint approach

2.1 Basic choices

Our method deals with independent solids linked by geometric constraints, expresses the action of

these constraints by corrections to the displacements, and solves the equations by decoupling the action of the constraints. Let us explain these choices.

2.1.1 For the models

Armstrong and Green (1985) represented articulated solids with tree-like structures, Barzel and Barr (1988) considered sets of independent solid components connected by contraints, and van Oberveld (1991) used only punctual masses.
We have chosen the second approach. Considering solids connected by geometric constraints is a very general way of modeling complex structures (these structures may contain closed loops, and even if they do not, there is no reason why one of the solids should play a particular role, such as the root of a tree). In addition, and contrary to van Overveld (1991), we prefer to use solid mechanics rather than expressing rigidity with an extra constraint between elementary masses.

2.1.2 For the way of expressing the action of constraints

Barzel and Barr (1988) treated geometric constraints by computing the reaction forces maintaining them. Then, these forces were integrated in the equations of motion. Van Overveld (1990, 1991) preferred to correct the momenta of the solids that were integrated to find new positions.
Our approach is different. We still want to find a motion correction that takes constraints into account. Nevertheless, we prefer to express our algorithm directly in terms of *adjusting displacements*. This point of view, which avoids the explicit computation of constraint forces, yields more efficent and more general algorithms:

- In an animation system based on dynamics, our corrections take into account the physical models of the solids. They are somewhat equivalent to the addition of some constraint forces, the values of which could be derived easily at each time step (but they are useless; directly adjusting displacements is more efficient).
- The avoidance of an explicit computation of forces gives a much more general aspect to our constraint processing. In fact, the algorithm would still work in an animation system with no physical model, for instance, in a system based on inverse kinematics, or on behavioral approaches. Simply, our corrections would no longer be based on physical data, but on parameters controlled by the user.

2.1.3 For the way of solving the equations

Two main approaches have been used to maintain constraints between independent components:

- The solution of a global system of coupled differential equations, each one associated with a constraint. This kind of approach was used by Barzel and Barr (1988).
- The decoupling of the action of constraints and of the other external actions. To do this, the components are first animated as if they were independent. Then, the constraints are taken into account by associating a set of corrections with each of them and by using a combination of the corrections with respect to each component. This process is iterated until all the corrections have become very small.

Van Overveld (1990; 1991) claimed that the latter approach avoids the heavy and complex computations associated with the former and that it offers a sufficient physical credibility, provided that the first-order linear and angular momenta are conserved during the constraint processing. We have chosen to experiment with a technique of that kind that can be implemented and tested very easily.

2.2 The animation algorithm for displacement constraints

In the remainder of this paper, all points, vectors and matrices are expressed in worldfixed coordinates.
The general algorithm for animation under constraints is directly derived from our three basic choices. To compute the vew positions and orientations of the solids at time $t+dt$ from a configuration at time t that satisfies the constraints:

1. Compute the new positions and orientations of the solids as if they were independent. In an animation system based on dynamics, this is done by integrating the Euler equations of motion (Goldstein 1983) over time for all the externally applied

forces and toques that are not due to constraints. For each solid:

$$\vec{v}(t+dt)=\vec{v}(t)+\frac{\sum \vec{F}(t)}{m}dt \quad (1)$$

$$\vec{\omega}(t+dt)=\vec{\omega}(t)+J^{-1}(\sum \vec{\mathcal{M}}(t)-\vec{\omega}(t)\wedge J\vec{\omega}(t))\,dt \quad (2)$$

$$\vec{x}(t+dt)=\vec{x}(t)+\vec{v}(t+dt)\,dt \quad (3)$$

$$R(t+dt)=(I+dt\,\tilde{\omega}(t+dt))\,R(t) \quad (4)$$

where m and J are the mass and tensor of inertia of the solid, respectively, $\vec{F}$ represents the external forces and $\vec{\mathcal{M}}$, the external torques with respect to the body's center of mass; $\vec{v}$ and $\vec{\omega}$ are the linear and angular speed vectors, respectively; $\tilde{\omega}$ is the rotation speed matrix "dual" of $\vec{\omega}$ (characterized by: $\forall \vec{a}\,\tilde{\omega}\vec{a}=\vec{\omega}\wedge\vec{a}$); $\vec{x}$ is the translation vector and R, the orientation matrix defining the current configuration of the solid.

This integration scheme for Euler equations of motion uses rotation vectors for small rotations during dt, which is a second-order approximation. If the rotations are larger than a given limit, the system goes back in time, and the time step is reduced. Furthermore, the matrix $R(t+dt)$ must be reorthogonalized just after using Eq. 4.

2. Take the constraints into account by slightly displacing the solids during a series of iterations (these displacements include small rotations and small translations). Because the constraints were satisfied at time t, just before the free motion during dt, the corrections reestablishing a given constraint cannot be larger than the sum of the constrained displacements of the solids during dt, so the formalism of the rotation vectors is also used to define the small rotations due to constraints. Sections 3 and 4, which deal with various kinds of constraints, detail the way we associate small displacements to each of them, and the way these displacements are combined when a solid is subjected to several constraints simultaneously.

Stop when all the resulting corrections are smaller than a specified limit, or when a maximum number of iterations is reached. Limiting the number of iterations avoids deadlocks when the system is over-constrained. When this maximum is reached, the system continues with the animation, thus introducing a configuration that does not exactly satisfy the constraints. A warning is issued, and the user can choose either to continue with the animation (for instance, if the problem is temporary, and the divergences are not too large) or to stop the animation and modify the set of constraints.

3. Adjust the linear and angular speeds of the solids by considering the positions and orientations that they have effectively reached during a time step (the corrections due to constraints must be taken into account in the kinematics of movement, as if we had added constraint forces):

$$\vec{v}(t+dt)=(\vec{x}(t+dt)-(\vec{x}(t))/dt$$

$$\tilde{\omega}(t+dt)=(R(t+dt)\,R^{-1}(t)-I)/dt$$

2.3 Automatic search of valid initial positions

The provision of an easy way to build models is very important in an animation system. In our framework of solids subjected to constraints, it would be very difficult to find valid initial positions of the objects manually. Please note that we really need such positions to start using our animation algorithm. Indeed, if the constraints were too far from being satisfied in the initial configuration, corrections of displacements during a small time interval that are too large would add extremely large values to the initial speeds.

Fortunately, a method for the automatic building of initial positions that satisfies the constraints can be derived from our approach easily. We just iterate our usual treatment of constraints (step 2 of the animation algorithm) from the arbitrary positions given by the user, but without updating the speeds of the solids when a solution is found.

During this process, the displacements computed at each time step must not be too large, because rotations are again expressed with rotation vectors. To keep the integration errors at an acceptable level, the convergence speed must not be too large (we will see in Sect. 3 that this speed can be controlled by the user, through a parameter α). As a larger number of iterations is needed to reach a valid configuration based on arbitrary positions of the solids, the maximum number of iterations is larger than that of an animation process.

3 "Point to point" constraints

This section describes the method of correcting displacements within the framework of the basic "point to point" constraints. Section 4 will extend

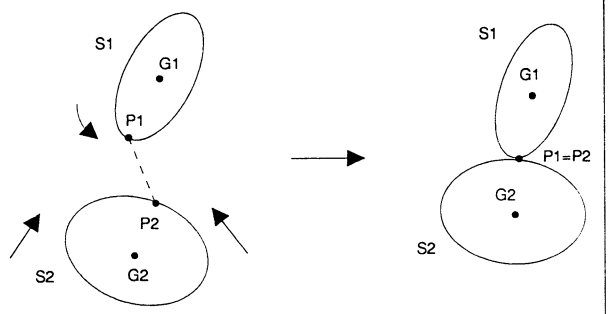


Fig. 1. A "point to point" constraint between two solids

the results to more complex constraints between solids.

With the same terminology as that of Barzel and Barr (1988), a "point to point" constraint between two solids S_1 and S_2 creates a joint with three degrees of freedom in rotation by making two points, P_1 of S_1 and P_2 of S_2, coincide during the overall movement (Fig. 1).

To maintain a set of "point to point" constraints during an animation sequence, two questions must be answered: How can we find reasonable displacements to satisfy a given constraint? How can we combine these displacements for a solid subjected to several constraints simultaneously?

3.1 Displacements associated with a "point to point" constraint

Satisfying a given "point to point" constraint by translating solids is always possible, but used alone, it would produce unrealistic behaviours. In consequence, the displacements we associate with a constraint must include both rotation and translation.

As we said previously, it seems natural to use the physical models of the solids (when we have them) while correcting their displacements. Indeed, the heaviest object must move less than the lightest. In addition, the relative amounts of rotation and translation applied to each solid must correspond to its mass and tensor of inertia.

3.1.1 Finding the right amount of rotation versus translation

To take the physical parameters of the solids into account while defining small rotations and small translations, we compare the action of a "point to point" constraint and the action of a piece of rubber that would pull point P_1 to point P_2 (Fig. 1).

Let J_1 and J_2 be the tensors of inertia of the solids at time t expressed in world-fixed coordinates; m_1 and m_2, their masses; and G_1 and G_2, their centers of mass.

A rubber of stiffness k connecting P_1 to P_2 would produce on S_1 and on S_2 the instantaneous forces and torques:

$$\vec{F}_1 = k(P_2 - P_1) \tag{5}$$

$$\mathcal{M}_{G1}(P_1, \vec{F}_1) = (P_1 - G_1) \wedge \vec{F}_1 \tag{6}$$

$$\vec{F}_2 = -\vec{F}_1 \tag{7}$$

$$\mathcal{M}_{G2}(P_2, \vec{F}_2) = (P_2 - G_2) \wedge \vec{F}_2 \tag{8}$$

If we want to consider the lasting action of this rubber over time, these values for forces and torques should be integrated in Euler equations of motion to determine the resulting displacements. This would lead to complex equations (where parameters P_i, G_i and J_i, although constant in the local coordinate system, would be functions of time because they are expressed in world-fixed coordinates) closely related to those used by Barzel and Barr (1988). (Note that Barzel does not decouple the action of individual constraints while solving the equations.) To generate animations at interactive rates, more approximations must be made.

To determine an expression for the displacements associated with a constraints, we integrate the action of the rubber during a small time interval Δt *that is considered to be small enough to enable the use of finite differences.* This is equivalent to neglecting the motion of the solids during Δt. This approximation is clearly justifed because the displacements that we are looking for will be smaller or equal than the displacements that produced the violation of the constraints during the dt for which the use of a finite difference integration of Euler equations of motion was justified.

The extra terms added by the force (Eq. 5) and the torque (Eq. 6) to the first solid's equation of motion (Eqs. 1 and 2) are:

$$\Delta\vec{v}_1 = \frac{k\Delta t}{m_1}(P_2 - P_1) \tag{9}$$

$$\Delta\vec{\omega}_1 = \Delta t\, J_1^{-1} k(P_1 - G_1) \wedge (P_2 - P_1) \tag{10}$$

(similar expressions are valid for the second solid). These extra amounts of linear speed and rotation speed produce the extra displacements:

$$\Delta\vec{R}_1 = J_1^{-1} \frac{k\Delta t^2}{2}(P_1 - G_1) \wedge (P_2 - P_1) \tag{11}$$

$$\Delta\vec{x}_1 = \frac{k\Delta t^2}{2m_1}(P_2 - P_1). \tag{12}$$

These expressions provide us with the relative values of the rotation and translation that correspond to the mass and the tensor of inertia of the solid. More precisely, the amount of rotation can be deduced from the amount of translation of the solid:

$$\Delta\vec{R}_1 = m_1 J_1^{-1}(P_1 - G_1) \wedge \Delta\vec{x}_1 \tag{13}$$

In addition, the rotations and the translations computed with this method for the two solids are derived from the action of the opposite forces and torques (Eqs. 5–8), so that they conserve the first-order linear and angular momenta[5] of the system (S_1, S_2).

3.1.2 Scheme for applying the small rotations and small translations

We want to use Eqs. 11 and 12 to define the small rotations and translations associated with the constraints. Of course, any value can be taken for the rubber's stiffness k, so these expressions can be rewritten:

$$\Delta\vec{R}_1 = \alpha J_1^{-1}(P_1 - G_1) \wedge (P_2 - P_1) \tag{14}$$

$$\Delta\vec{x}_1 = \frac{\alpha}{m1}(P_2 - P_1) \tag{15}$$

where α is a scalar parameter that can be fixed by the user and that is related to the convergence speed and to the correctness of the motion. Indeed, taking a very small value for α during the iterative constraint processing would correspond to a precise integration over time of the action of the rubber between constrained points, but this small value would also lead to a low convergence speed, so a sompromise must be found.

In practice, a different value for α can be chosen for each constraint between a pair of solids, and the rotations and translations are computed and applied in a sequential way. When a single constraint is specified, the constraint processing scheme consists in iterating:

1. Apply the small rotation of Eq. 14
2. Apply the small translation of Eq. 15

until the constraint is verified (Sect. 3.2 will extend this scheme to multiple-constraints situations).

3.1.3 A possible criteria for choosing α:

To solve the simplest situations very quickly, the best method would be to find a computation scheme that satisfies the constraint after the first iteration of the constraint processing, provided that no other constraint is applied to the solids. This gives us a good criterion for choosing α, but obliges us to modify the iterations scheme slightly. Indeed, "point to point" constraints cannot always be satisfied by pure rotation, but are easy to satisfy exactly with an adequate translation of the solids. If we wanted to have:

$$P_1 + \Delta\overrightarrow{x_1} = P_2 + \Delta\overrightarrow{x_2} \tag{16}$$

taking the following value for α would be sufficient:

$$\alpha = \frac{m_1 m_2}{m_1 + m_2} \tag{17}$$

In practice, we want to combine this translation (which gives an exact result) with a small rotation. To obtain an exact solution for the constraint in one iteration, we use a new computation scheme:

1. Apply the small rotations:

$$\overrightarrow{\Delta R_1} = \frac{m_1 m_2}{m_1 + m_2} J_1^{-1}(P_1 - G_1) \wedge (P_2 - P_1)$$

$$\overrightarrow{\Delta R_2} = \frac{m_1 m_2}{m_1 + m_2} J_2^{-1}(P_2 - G_2) \wedge (P_1 - P_2) \tag{18}$$

2. Then, from the positions P_1', P_2' of the points *after rotation*, compute and apply the small translations exactly satisfying the constraint:

[5] *Let* $G = m_1 G_1 + m_2 G_2$ be the center of mass of the system (S_1, S_2). The linear momentum of the system is $(m_1 + m_2)\vec{v}_G = m_1\vec{v}_1 + m_2\vec{v}_2$, and the angular momentum is $J\vec{\omega}_G$, where the inertia tensor J is computed with respect to G. These quantities are conserved by the action of internal opposite forces applied on S_1 and S_2, when no external force or torque is acting on the system (Goldstein 1983).

$$\overrightarrow{\Delta x_1} = \frac{m_2}{m_1 + m_2}(P_2' - P_1')$$

$$\overrightarrow{\Delta x_2} = \frac{m_1}{m_1 + m_2}(P_1' - P_2') \quad (19)$$

Note: This algorithm modifies the amounts of translaton versus rotation slightly, because the translation vectors are no longer computed from the same positons of P_1 and P_2 as those of the rotation vectors. Then, although it takes the physical data into account, the second scheme is considered a good heuristic approach rather than a close approximation of the exact motion. Nevertheless, it gives very satisfactory results in practice (motion computed with this method "seems realistic") and leads to very fast convergence rates for constraint processing (for the "monocycle" example of Sect. 5.3, this second scheme for constraint processing converges most of the time with less than six iterations per frame).

3.1.4 Extension to animation systems using no physical data

In an animation system with no physical model, constraints speed can be iteratively maintained with a good convergence by using the second computation scheme, where physical data are replaced by intuitive user's parameters:

- The factor $m_2/(m_1+m_2)$ in the translation of S_1 (Eq. 19) is proportional to the amount of translation of S_1 compared with that of S_2. It can be replaced easily by a user parameter μ_{12} associated with a given constraint, verifying $0 < \mu_{12} < 1$ and specifying the "ability to move" of one solid relative to the other (the factor for translation of S_2 would be $\mu_{21} = 1 - \mu_{12}$).
- The expression $m_1 J_1^{-1}$, which can be factorized together with μ_{12} in the S_1 rotation (Eq. 18), controls the amount of rotation of S_1 versus translation. This matricial expression can be replaced with a positive scalar parameter[6] associated with each solid provided that its "ability to rotate" is specified.

[6] Replacing the inertia tensor with a scalar is equivalent to considering a spherical mass distribution.

3.2 Combining displacements due to individual constraints

In practice, several constraints can be applied to a solid simultaneously, so that rotations and translations due to particular constraints must be combined. To find the displacement due to a given set of constraints, we use a linear combination of the particular displacements. Although rotations are not commutative in general, we have computed "small rotations" approximated by rotation vectors. One of the advantages of this formulation is that small rotations can be combined in a commutative way, exactly as for translations: the resulting rotation vector can be defined by a weight sum of rotation vectors due to individual constraints.
At each iteration of the constraint processing:

- We first compute and combine the small rotations.
- Then, we do the same thing for the translations.

Depending on which scheme is used, the translations can be computed from the orientations of the solids either before or after rotation.
Our way of combining rotations and translations must preserve the property of conservation of the first-order momenta obtained when a single. A sum of displacements would work [this solution, which corresponds to summing the constraints forces applied on each solid, was used by van Overveld (1990; 1991)]. Nevertheless, this method would lead to divergences in some cases: suppose that four solids S_0 to S_3(S_1, S_2 and S_3 being located in the same place) are linked by three "point to point" constraints between G_0 and each of G_1, G_2, G_3 (Fig. 2). With this configuration, summing the three translations computed for S_0 (the mass of which is assumed to be half the mass of the other solids) and displacing S_1, S_2 and S_3

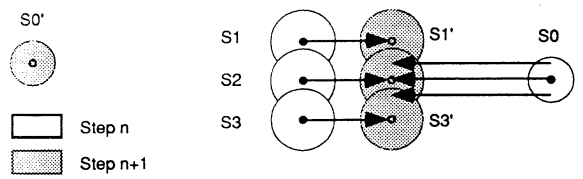


Fig. 2. For these solids, summing the translations associated with each constraint would move them farther away from a solution at each iteration

would move them farther and farther from a solution at each iteration of the constraint processing.

To avoid this problem, we weight the displacements affected by the solids before summing them. In order to conserve the first-order momenta, the same weighting factor must be used for couples of displacements due to the same constraint. If S_i and S_j are two objects linked by a given constraint, we weight the associated displacements with:

$$\frac{1}{\max(n_i, n_j)}$$

where n_i and n_j are the numbers of constraints applied to the solids, respectively.

Let us explain briefly why this factor works by studying more precisely the simple cases where constrained points P_i coincide with the G_i, as in Fig. 2 (in these cases, our algorithm only produces translations of the solids). In the following proof, the value $m_i m_j/(m_i+m_j)$ is used for the parameter α associated with the constraint between S_i and S_j. However, the same would be true at least for any other value smaller than m_i and m_j (the key point is that the coefficients λ_{ij} must be smaller than unity).

Let us call G_i^n the position of G_i after n iterations of the constraint processing, and C_i the set of indices j for which the solid S_j is constrained with S_i (by convention, $i \in C_i$). The weighted sum of translations gives:

$$G_i^{n+1} = \sum_{j \in C_i} \lambda_{ij} G_j^n \tag{20}$$

where:

$$\lambda_{ij} = \frac{m_i}{m_i+m_j} \frac{1}{\max(n_i, n_j)} \quad \text{if } j \in C_i - \{i\}$$

$$\lambda_{ii} = 1 - \sum_{j \in C_i - \{i\}} \lambda_{ij} \tag{21}$$

The coefficients λ_{ij} sum to one, and verify:

$$\forall i, \forall j \in C_i, \quad 0 < \lambda_{ij} < 1 \tag{22}$$

This is quite evident if $i \neq j$. To determine it for λ_{ii}, we just have to note that if $j \neq i$, $\lambda_{ij} < (1/n_i)$, and that n_i is the cardinal of the set of indices $C_i - \{i\}$.

Thus, with our choice for the weighting factors, G_i^{n+1} is a barycenter of the $(G_j^n)_{j \in C_i}$. Because of the strict inequalities of Eq. 22, it lies in the interior of the convex hull of these points, and the situation in Fig. 2 can no longer occur.

Note: A formal convergence proof of the constraint processing would be quite hard to formulate. It would involve both characterizing over-constrained situations (for which no solution can be found) and dealing with constrained systems with an infinity of valid solutions. Even for systems with exactly the right number of constraints, a formal proof taking the corrections in rotation and in translation into account would be considerably more involved. Nevertheless, we can note that the topology of the constraint graph does not appear in the equations nor in the resolution scheme. The partial proof described here works just the same way in a configuration with or without closed loops, and it appears to be the same for a more general proof. In practice, we have applied the displacement-constraint method in various situations with very different topologies for the graph of constrained solids. We had no problems with convergence, even in complex situations such as the one in Fig. 8.

4 Extensions to other constraints

This section presents heuristics that extend the "point to point" constraint method to more complex relationships between connected solids. First of all, we study two constraints that introduce a number of degrees of freedom in translation, and then we present angle and twist constraints, which provide a precise control of the rotations at hinges. Translation and rotation constraints can be used together, so that any number and combination of degrees of freedom can be chosen for each joint.

4.1 Translations at hinges, possibly user-restricted in scope

4.1.1 "Point to segment" constraints

These constraints give one degree of freedom in translation at joints, possibly user-restricted to a given interval. The user chooses two points P_1 and P_2 on the solids as before and defines a segment passing through P_2 by a unit vector $\vec{t}_2$ linked to S_2 and by a scalar interval $[\lambda_{\min}, \lambda_{\max}]$ (that can be $[-\infty, +\infty]$ if needed). Then, the constraint is:

$$P_1 = P_2 + \lambda \vec{t}_2 \quad \text{with} \quad \lambda \in [\lambda_{\min}, \lambda_{\max}].$$

Visual Computer

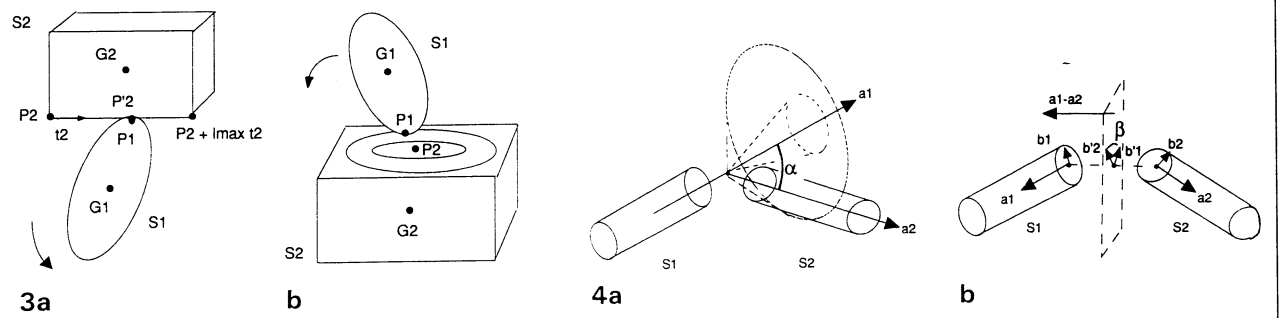


Fig. 3. a A "point to segment" constraint. **b** A "point to ring" constraint

Fig. 4. a Angle and **b** twist constraints between solids

The displacements of S_1 and S_2 associated with this constraint at each iteration of the constraint processing are computed by replacing P_2 in the equations that give the small rotation and the small translation with the point P_2' of the segment $[P_2+\lambda_{\min}\vec{t}_2, P_2+\lambda_{\max}\vec{t}_2]$ that is the closest to P_1. Thus, S_1 will be free to translate along the permissible segment of S_2 (Fig. 3a).

4.1.2 "Point to ring" constraints

The "point to ring" constraints allow a point P_1 of S_1 to translate in a plane linked to S_2, with a possible limitation to a disc or to a ring-shaped domain. The user defines the plane with $(P_2, \vec{n}_2)$ (normal vector), and gives an interval $[r_{\min}, r_{\max}]$ to limit the distance between P_1 and P_2 (Fig. 3b). If $r_{\min}=0$, the domain is a disc.
To compute the displacements associated with this constraint, P_2 is replaced in Eqs. 18 and 19 with the point P_2' of the ring-shaped domain that is the closest to P_1 (P_2' is the projection of P_1 onto the plane if this projection lies inside the ring; elsewhere, P_2' is the point of the ring that is the closest to the projection).

4.1.3 Other translation constraints

The movement of P_1 with respect to S_2 can be easily constrained to any other area defined with respect to S_2 (and of dimension 1, 2, or 3), as long as an efficient procedure is known to compute the nearest permissible point P_2'. In particular, "point to curve," "point to surface," "point in sphere" constraints can be useful.

4.2 Restricting rotations at hinges

To decrease the number of permissible rotations, or to limit their scopes, the user can combine the translation constraints defining a joint with angle and twist constraints.

4.2.1 Angle constraints

The user defines two unit vectors, $\vec{a}_1$ and $\vec{a}_2$, respectively linked to S_1 and S_2, and specifies an angle constraint:

$\widehat{\vec{a}_1\,\vec{a}_2} \in [a_{\min}, a_{\max}] \subseteq [0, \pi]$.

If $a_{\min}=a_{\max}$, this forces $\vec{a}_2$ to lie on the surface of a cone of axis $\vec{a}_1$. Otherwise, S_2 still has three degrees of freedom in rotation with respect to S_1, but $\vec{a}_2$ must stay in the difference of two cones, fixed with respect to S_1 (Fig. 4a).
To find the correction of displacements at a joint subjected to an angle constraint, we use the following heuristic: the rotation vectors $\Delta\vec{R}_1$ and $\Delta\vec{R}_2$ are first computed according to the main joint constraint (for instance "point to point," "point to segment," or "point to ring"). Then, if $\widehat{\vec{a}_1\,\vec{a}_2}$ does not lie in the given interval, rotations of the solids in the plane $(\vec{a}_1, \vec{a}_2)$ are added (we use their inertia J_1 and J_2 to conserve angular momenta). The translations are computed in the usual way.

4.2.2 Twist constraints

The user still gives $\vec{a}_1$ and $\vec{a}_2$ of S_1 and S_2 and defines two other vectors $\vec{b}_1$ and $\vec{b}_2$ of the solids, respectively perpendicular to $\vec{a}_1$ and to $\vec{a}_2$. Let $\vec{b}_1'$

and $\vec{b}'_2$ be the projections of $\vec{b}_1$ and $\vec{b}_2$ on the median plane, of normal $(\vec{a}_1 - \vec{a}_2)$ (Fig. 4). The user limits the twist by giving an angular interval $[b_{min}, b_{max}]$ for the angle between $\vec{b}'_1$ and $\vec{b}'_2$.
To correct the rotation associated with this joint, the solid S_1 (or S_2) is possibly rotated around $\vec{a}_1$ (or $\vec{a}_2$) according to the twist constraint (again, the inertia of the solids is used to conserve the angular momenta).

4.2.3 Combining angle and twist constraints

Angle and twist constraints often need to be combined, for instance, to define a complex hinge in an articulated body. Then, the two constraints can be specified with respect to the same vectors $\vec{a}_1$ and $\vec{a}_2$. To correct the rotation vectors associated with the joint, rotations in the plane $(\vec{a}_1, \vec{a}_2)$ are added first in order to satisfy the angle constraint. Then rotations around $\vec{a}_i$ are computed according to the twist constraint (this second rotation does not modify the angle $\widehat{\vec{a}_1 \vec{a}_2}$).
Angle and twist constraints can also be very useful in situations where there is no hinge and no translation constraint between the solids. The "monocycle" example of Sect. 5.3 demonstrates this point.

Note: The complete suppression of the rotations between two solids may be useful. This can be done easily by choosing $(b_{min} = b_{max})$ and $(a_{min} = a_{max} = 0)$ in an angle and twist constraint.

5 Implementation and results

5.1 Animation framework

We have implemented the displacement-constraints method as an extension of the animation system described by Gascuel et al. (1991)[7]. This system is based on solids mechanics and enables one to coat objects with deformable material.
More precisely, a solid is structured in:

- A "kernel" that contains the data needed for animating its local coordinate system. Kernels can be immobile, follow a predefined trajectory, or be dynamically animated by the integration of Euler equations of motion over time.
- The material, rigid or deformable, that constitutes the object (defined in the local coordinate system). A detailed description of our model for deformable material can be found in Gascuel et al. (1991). Its main features are the simple and efficient control of deformations and the automatic response to collisions that it provides.
- The "skin" surface controlled by the deformations of underlying material. A chain of objects can be coated with the same continuous skin, which is useful for solids connected by hinges [we often use bicubic spline surfaces (Fig. 5)].

5.2 User interface

In the current implementation, we use a description language with a set of simple key words to define objects and constraints. A full example description file is given in Appendix A. The syntax for describing the set of constraints connecting a pair of solids is the following:

```
constraint
    object1 name;     (default to world)
    object2 name;
    hinge   P1 P2;    (default to none)
    axial   u1^N {min λ1} {max λ2};
                (default to no translation)
    planar  n1^N {min ρ1} {max ρ2};
                (default to no translation)
    angle   a1 a2 {min α1} {max α2};
                (default to no angle constraint)
    twist   b1 b2 {min β1} {max β2};
                (default to no twist constraint)
end
```

Many situations require constraints that vary over time. Specifying these dynamic constraints can be done easily with our description format by specifying in which time interval each constraint is valid.
Describing contraints with this language has been a good way of experimenting very rapidly with various kinds of constraints. In the future, we plan to implement a graphic interface that will take advantage of the intuitive geometrical meaning of the parameters and options (hinge, axis, plane, ...).

[7] In the previous version of this system, articulated objects were animated with Armstrong and Green's technique (1985). This method was quite efficient, but limited to hinges with three degrees of freedom in rotation, and none in translation. In addition, angle and twist constraints could not be specified.

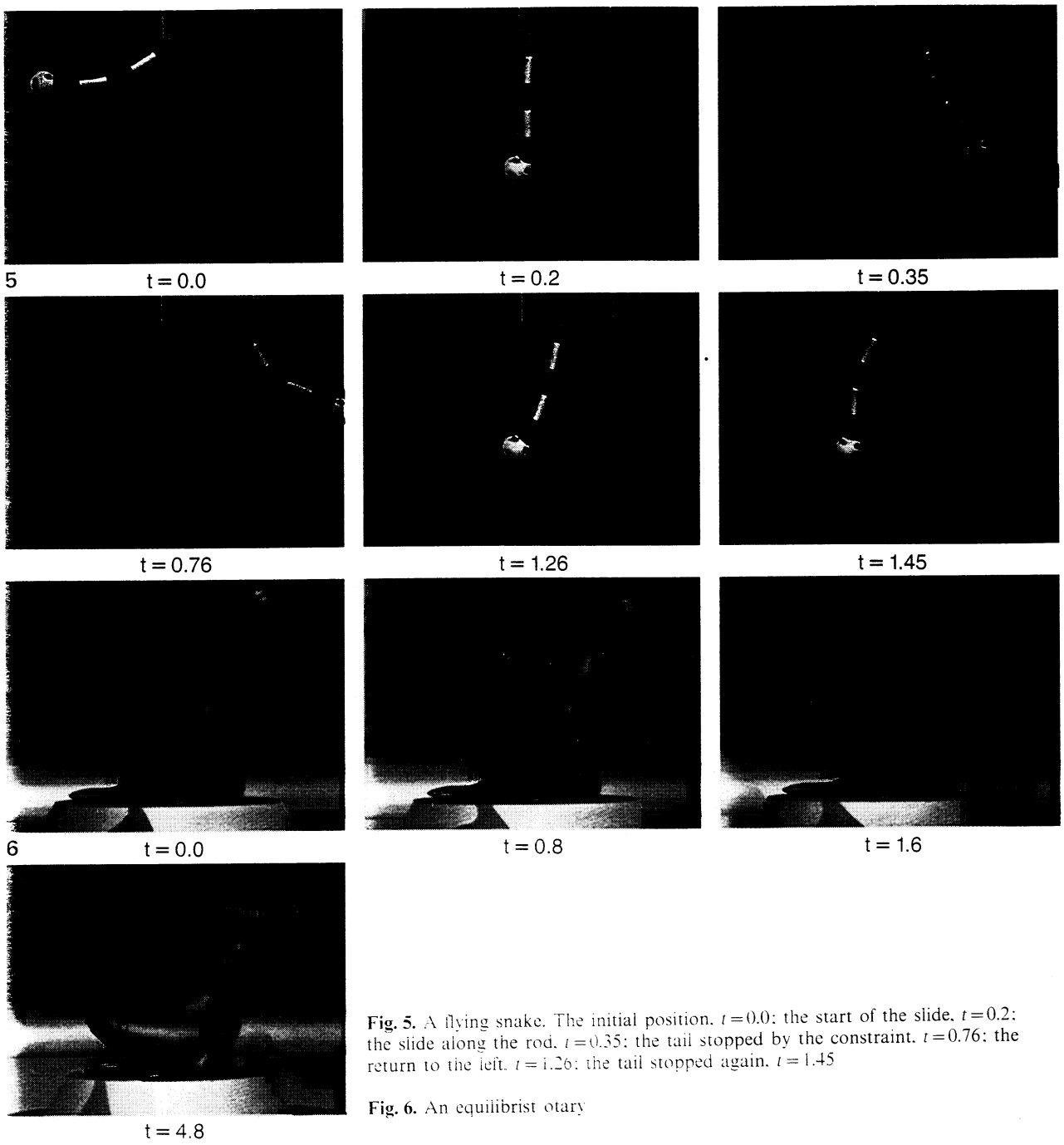


Fig. 5. A flying snake. The initial position, $t=0.0$; the start of the slide, $t=0.2$; the slide along the rod, $t=0.35$; the tail stopped by the constraint, $t=0.76$; the return to the left, $t=1.26$; the tail stopped again, $t=1.45$

Fig. 6. An equilibrist otary

5.3 *Samples of animations*

5.3.1 Flying snake

The snake animated in Fig. 5 is composed of six deformable solids connected by five "point to point" constraints and covered by the same continuous skin. A "point to segment" constraint allows the tail of the snake to slide along a rigid rod. Please note the complex motion of the snake: the tail slides first to the right because of the combined action of gravity and of the constraints on the snake. Then it is stopped by the "point to segment" constraint, which makes the snake swing. Finally,

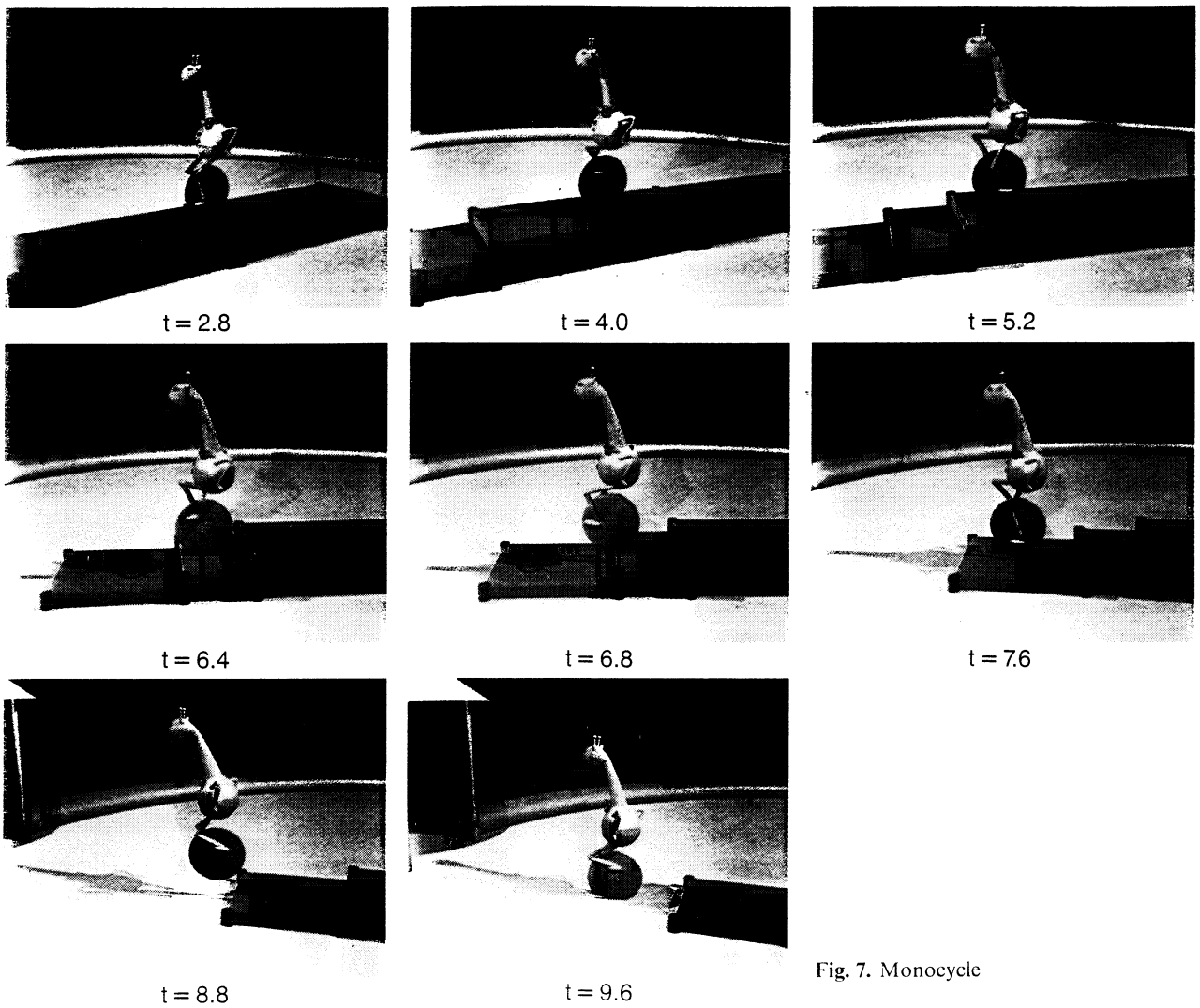


Fig. 7. Monocycle

the snake flies back to the left, which makes the tail slide again.

5.3.2 Equilibrist otary

This animation illustrates the use of a "point to ring" constraint. In this particular case, the ring is a disc: a deformable ball is constrained to stay on a tray held by an otary.
The otary is composed of three deformable components, the head, the body, and the tail, connected by "point to point" angle and twist constraints and covered by a continuous skin. The tray is maintained on the otary's nose by a "point to point" constraint. An angle constraint with the world-fixed coordinate system keeps it almost horizontal. During the animation, various external forces and torques are applied to the otary's head and tail to make the system move.

5.3.3 Monocycle

This animation involves six "point to point" and angle constraints, two "point to point"-only constraints, and three angle-only constraints. The graph of constrained objects, depicted in Fig. 8, includes several closed loops.
The animation sequence shown in Fig. 7 has been generated simply by applying a horizontal force to the character's body. A "point to point" con-

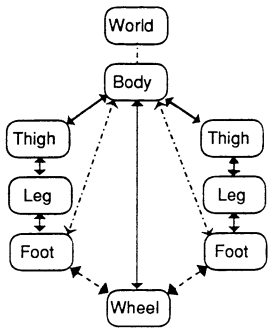


Fig. 8. The graph of constrained objects used for the "monocycle" sequence includes several closed loops
◀----▶ Point to point only.
◀——▶ Point to point +Angle,
◀·----▶ Angle only

straint between the body and the center of the wheel generates the motion of the wheel, but because of its interactions with the floor, the wheel rolls rather than slides: the wheel is coated with deformable material, and the model used provides an automatic response to collisions and to lasting contacts (Gascuel et al. 1991), so that the rotation of the wheel is automatically generated by friction with the floor. This rotation is transferred to the legs through the action of the constraints. More precisely, the feet are constrained to stay on the wheel, and an angle constraint keeps them almost horizontal. "Point to point" and angle constraints connect the different components of the legs. Finally, an angle constraint between the world-fixed coordinate system and the character's body prevents the monocyclist from falling over the floor.
More complex than the previous ones, this animation is still interactive and did not raise a convergence problem. The full description file used to specify the constraints is given in Appendix A.

6 Conclusion

We have presented an integrated set of methods for building and animating systems of solids connected by geometric constraints. Our techniques, based on iterative corrections of displacements, are easy to implement and enable the simulation a large variety of objects and of motions. Any number of translations or rotations can be allowed at each joint between solids. Each degree of freedom can be limited in scope by the user. The method still works when the graph of constrained objects contains closed loops.

As in Barzel and Barr (1988), an automatic technique for assembling the objects before animation is provided. The construction of animation models, very hard to achieve manually, is significantly simplified. In modeling systems, our technique would facilitate the independent modeling of the different components of a complex scene. This components would be positioned without requiring the use of physical modeling for the solids by geometric constraints.
Displacement constraints could be integrated into various kinds of animation systems (for instance, in systems based on inverse kinematics, or on behavioral approaches). In the present implementation, the method is inserted in a dynamic simulation system, where it provides a good compromise between the efficiency of the computations and the "correctness" of the results: corrections associated with the constraints take the physical data into account, and first-order momenta are conserved during the constraint processing.
We are currently studying methods of extending displacement constraints to "soft constraints" between solids. With soft constraints the user will be able to specify preferential angular or linear domains for the relative positions of objects at hinges without restricting the motion to specific predefined boundaries. The solids will tend to satify the soft constraints if they are not subjected to forces that are too large, and if their accelerations are not too important.
We are interested in the ability of soft constraints to make the behavior of solid systems more flexible and we plan to use them as a compromise in over-constrained situations.

References

Armstrong W, Green M (1985) The dynamics of articulated rigid bodies for purposes of animation. Graph Inerface 85, Montreal, pp 407–415
Barzel R, Barr A (1988) A modeling system based on dynamic constraints. Comput Graph 22:179–188
Featherstone R (1983) The calculation of robot motion using articulated-body inertias. Int J Robotics Res 2:13–30
Gascuel MP, Verroust A, Puech C (1991) A modeling system for complex deformable bodies suited to animation and collision processing. J Visualization Comput Animation, 2:82–91. A shorter version of this paper appeared in Graphics Interface 91
Goldstein H (1983) Classical Mechanics, 2nd edn. Addison Wesley, Reading, Mass

Isaacs PM, Cohen UF (1987) Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. Comput Graph 21:215–224
Isaacs PM, Cohen UF (1988) Mixed method for complex kinematic constraints in dynamic figure animation. Visual Comput 2:296–305
Lathrop RH (1986) Constrained (closed-loop) robot simulation by local constraint propagation. IEEE Int Conf Robotics Automation San Francisco, pp 689–694
Overveld C van (1990) A technique for motion specification in computer animation. Visual Comput 6:106–116
Overveld C van (1991) An iterative approach to dynamic simulation of 3-D rigid-body motions for real-time interactive computer animation. Visual Comput 7:29–38
Verroust A (1990) Etude de problèmes liés à la définition, la visualisation et l'animation d'objets complexes en informatique graphique. Thèse d'état, Université Paris XI, France
Wilhelms J, Barsky B (1985) Using dynamic analysis to animate articulated bodies as humans and robots. Graph Interface 85, Montreal, pp 97–104

Appendix A: The description file used for the monocycle

This is the description file for all the constraints involved in the monocycle animation sequence. The objects linked by the constraints are the fixed external world, the wheel, the character's body, and the legs which are constituted of `left-thigh`, `right-thigh`, `left-leg`, `right-leg`, `left-foot`, `right-foot`. (in this description *Oy* is the vertical axis).

```
/* Constraint between the wheel and the
character's body */
constraint objectl body; object2 wheel;
hinge 0 -2.0 0 0 0 0;
angle 1 0 0 1 0 0 max 0; end

/* Angle constraint preventing the character
from falling */
constraint object2 body; /* default objectl:
world-fixed coordinate system */
angle 0 1 0 0 1 0 max 0.066; end

/* Constraints along the left leg */
constraint objectl body; object2 left-thigh;
hinge 0.4 -0.6 0  0 0 0;
angle 1 0 0 1 0 0 max 0; end

contraint objectl left-thigh;
object2 left-leg; hinge 0 0 1  0 0 0;
angle 1 0 0 1 0 0 max 0; end

constraint objectl left-leg;
object2 left-foot; hinge 0 0 1.4  0 0 -0.1;
angle 1 0 0 1 0 0 max 0; end

constraint objectl wheel; object2 left-foot;
hinge 0.4 0 -0.5 0 0 0; end

constraint objectl body; object2 left-foot;
angle 0 0 1 0 0 1 max .1; end

/* Constraints along the right leg */
constraint objectl body;
object2 right-thigh;
hinge -.4 -0.6 0  0 0 0;
angle 1 0 0 1 0 0 max 0.04; end

constraint objectl right-thigh;
object2 right-leg; hinge 0 0 1  0 0 0;
angle 1 0 0 1 0 0 max 0; end

constraint objectl right-leg;
object2 right-foot; hinge 0 0 1.4  0 0 -0.1;
angle 1 0 0 1 0 0 max 9; end

constraint objectl wheel;
object2 right-foot;
hinge -0.4 0 0.5  0 0 0; end

constraint objectl body; object2 right-foot;
angle 0 0 1 0 0 1 max .1; end
```

JEAN-DOMINIQUE GASCUEL graduated from the Ecole Normale Supérieure (mathematics and computer science) in 1989. He received the degree of PhD in 1991 from the University of Orsay, France, for his work on the design of a Hopfield neural network integrated circuit. He has been both an Assistant Professor of Computer Science at the Ecole Polytechnique, France, and a researcher at the Centre National de la Recherche Scientifique (CNRS) since October 1991. He has worked on the implementation of various neural architectures, and he has been involved in computer graphics and more particularly in computer animation since June 1991.

MARIE-PAULE GASCUEL is an Assistant Professor of Computer Science at the Ecole Normale Supérieure (ENS), Paris, France. A graduate from the ENS, where she studied both mathematics and computer science, she received the degree of PhD in Computer Science from the University of Paris-sud/Orsay, France, in October 1990. Her research interests include computer-aided geometric design, implicit surfaces, computer animation of linked figures, and physically based animation.

Chapitre 3

An implicit formulation for precise contact modeling

Ce travail sur la modélisation fine du contact entre objets déformables, grâce à des surfaces implicites, a fait l'objet d'une présentation à la conférence internationale *SIGGRAPH'93*, ainsi que d'une présentation en France :

[Gas93] An Implicit Formulation for Precise Contact Modeling between Flexible Solids. Marie-Paule Gascuel. In *SIGGRAPH'93* (Anaheim, California, August 1993). Computer Graphics, pages 313–320, août 1993.

[Gas93F] Un modèle implicite pour une modélisation fine des contacts entre objets déformables. Dans les actes des *Journées AFIG-GROPLAN 1993*, Bordeaux, décembre 1993.

Résumé

Ce papier présente un modèle déformable implicite, construit à partir d'isosurfaces de champs potentiel engendrés par des squelettes. Il fournit une représentation élégante et unifiée des paramètres géométriques de objets (forme et déformations) et de leurs propriétés physique comme la raideur. Le modèle est tout particulièrement conçu pour améliorer le traitement des collisions et des contacts entre objets déformables. En particulier, il engendre et maintient des surfaces de contact exactes entre objets en interaction.

An Implicit Formulation for Precise Contact Modeling between Flexible Solids

Marie-Paule Gascuel*
iMAGIS, LIENS, CNRS URA 1327
Ecole Normale Supérieure, 45 rue d'Ulm
75005 Paris, France

Abstract

This paper presents an implicit deformable model, based on isosurfaces of potential fields generated by skeletons, that provides elegant and unified formulations for both geometric parameters such as shape or deformation and physical properties such as rigidity. The model is especially designed to improve collision and contact processing for non-rigid objects. In particular, it generates and maintains exact contact surfaces during interactions.

Keywords: animation, simulation, deformation, implicit surface, collision detection, collision response.

1 Introduction

Dynamic animation systems based on simplified physical laws have drawn a lot of attention during the past few years. One of the reasons why they seem so attractive is their ability to respond automatically to collisions. Nevertheless, contrary to rigid solid animation where complete analytical solutions have been found [1], modeling interactions between deformable objects still remains a challenge. In particular, none of the models proposed up to now generates an exact contact surface between interacting flexible solids.

This paper presents a new, continuous model for deformable material based on an implicit formulation which unifies the description of geometry and of physical properties of solids. Well adapted to the simulation of local deformations, the model is especially designed to improve collision and contact processing for non-rigid objects. In addition to an efficient collision detection mechanism, it generates and maintains exact contact surfaces during interactions. These surfaces are then used for the calculation of reaction forces.

Compact, efficient, easy to implement and to control, our implicit deformable model would be a particularly convenient tool in character animation where locally deformable flesh must be simulated.

*From september: iMAGIS, IMAG, BP53X 3841 Grenoble Cedex.

1.1 Previous approaches

Flexible models in Computer Graphics result from either nodal approaches (which include finite elements [6], finite differences [12, 13], and systems using elementary masses [8, 7]) or global approaches [10, 14]. The latter optimize the animation by approximating deformations by particular classes of global transformations. Well adapted to the animation of homogeneous blocks of elastic material, they would not, however, be convenient to use when simulating a material subject to local deformations (a sponge for instance), or when modeling non homogeneous complex objects like those used in character animation (typically, deformable coating over rigid skeletons).

Collisions between flexible objects are a complex phenomenon. In particular, they are not instantaneous and do not conserve energy. Among the solutions used to cope with this problem in Computer Graphics, penalty methods [9] are probably the most widely spread. They don't generate any contact surface between interacting flexible solids but use instead the amount of local interpenetration to find a force that pushes the objects apart. A different solution consists in using the relative stiffnesses of solids to find correct deformed shapes in contact situations. Here, response is computed by integrating deformation forces within the contact areas. But combined with a deformable model based on spline surfaces controlled by discrete spring systems [5], this method does not generate exact contact surfaces. A third approach [2] extends the analytical interaction processing used for rigid solids [1] to a global deformable model [14]. Contact surfaces are approximated by discrete sets of contact points which, as the authors emphasize, is somewhat unsatisfactory.

In all these methods, the lack of a contact surface between interacting flexible solids generates local interpenetration and imperfectly deformed shapes. The extent of these artifacts is exacerbated by the lasting quality of soft collisions, and forbids any correct evaluation of reaction forces.

1.2 Overview

This paper presents a new deformable model which improves interaction processing for flexible solids. Our main point is the use of isopotential implicit surfaces generated by "skeletons" to model the objects. Developed up to now as a tool

 ACM-0-89791-601-8/93/008/0313 $01.50

r free form modeling [4, 3], this formalism has not been ıed as a way to model physical properties[1]. It leads to concise formulation for both geometric parameters, such shape and deformation, and physical properties, such as ıidity and elastic behavior. The deformable model is conıuous and provides easy modeling of local deformations. The associated method for collision detection and response an improved version of [5]. The inside/outside functions ıociated with implicit solids greatly reduces the compu.tional cost of collision detection. The model generates ıd maintains exact contact surfaces between interacting obcts. Opposite compression forces are respectively applied ı the solids along contact surfaces, so a correct integration response forces can be calculated.

Section 2 describes the implicit deformable model, and exains how to design both homogeneous and non-homogeneıs flexible solids. The processing of interactions is detailed Section 3, including the particular cases of multiple colıions and of interactions with rigid solids. Section 4 disısses implementation. Section 5 focuses on the possibilities r future research opened by our method.

Implicit Deformable Solids

ur aim is to simulate damped material where deformations ıe to collisions remain local. Rather than considering the ıneral Lagrange equations of motion for non-rigid objects ıs in [12]), we use the same approximation as in [13, 5]: the .ass distribution of solids is considered to be constant, so ıotion is calculated using rigid body equations which lead ı a more efficiently computed animation. More precisely, ıformable solids are split into two layers:

- A rigid component which obeys the rigid body equations of motion. Its mass distribution corresponds to the object's rest shape.
- A deformable layer at rest relative to the rigid layer.

This section presents a new model for the deformable layer ıased on implicitly defined isopotential surfaces generated y skeletons. We first review the definition of these surfaces.

.1 Implicit surfaces

nplicit surfaces such as "distance surfaces" [4] and "conıolution surfaces" [3] allow the free form design of shapes ırough the manipulation of "skeletons" that generate poıntial fields. Very simple to define and to control, they ınstitute a good alternative to traditional implicit surfaces ıefined by analytical equations. An implicit surface S generted by a set of skeletons $S_i (i = 1..n)$ with associated "field ınctions" f_i is defined by:

$$S = \{P \in \Re^3 \ / \ f(P) = 1\} \text{ where } f(P) = \sum_{i=1}^{n} f_i(P)$$

[1] Flexible solids have been described by superquadrics [10, 11], anıher kind of implicit surface. Contrary to the approach developed ıre, the choice of an implicit geometric description of objects was ıt closely related to the way physical properties were modeled.

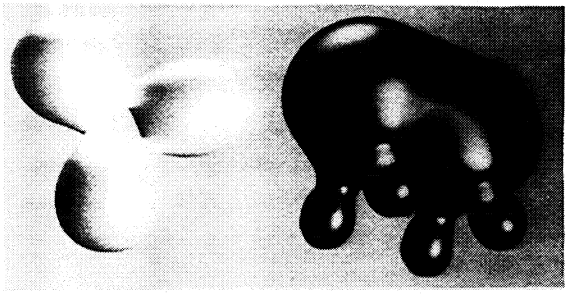


Figure 1: Isopotential objects generated by skeletons.

This surface surrounds the solid defined by $f(P) \geq 1$, which can have several disconnected components. Normal vectors are directed along the field's gradient.

The skeletons S_i can be any geometric primitive admitting a well defined distance function, such as: points, curves, parametric surfaces, or volumes. The field functions f_i are monotonically decreasing functions of the distance to the associated skeleton [4]. For convolution surfaces [3], they are given by integrals of exponential contributions from each point of the skeleton. In order to optimize the computations, these functions usually have a restricted scope of influence. Examples of isopotential surfaces are shown in Figure 1.

An implicit surface can easily be deformed by introducing a deformation term g in its implicit representation e.g. $f(P) + g(P) = 1$. In the remainder of this paper, only this type of deformations are considered.

2.2 Defining elastic material with potential fields

The method is based on the following observation: the set of points P satisfying $f(P) = 1$ (where f is the field function) is sufficient to define a surface. This set of points being fixed, the variation of f around the isosurface can be used to model physical properties. The next section explains how to express stiffness with field functions in a way which yields a very simple correspondence between applied forces and resulting deformations.

Correspondence between forces and deformations

A deformable model is defined by a correspondence between forces and deformations. In computer graphics, this correspondence has been given by both linear [13, 6, 14] and non-linear [12] elasticity. In non-linear models, the stiffness k is not only a function of the point P you consider, but may also depend on its current location inside the solid. The applied force during a displacement of P from $X_0 = (x_0, y_0, z_0)$ to $X(P) = (x(P), y(P), z(P))$ is:

$$R(P) = \int_{X_0}^{X(P)} k_P(Y)\, dY \qquad (1)$$

To improve generality, implicit deformable solids should be capable of exhibiting both linear and non-linear behaviors.

In practice, the correspondence between forces and deformations will be used during the collision process, to integrate the reaction forces colinear to normal vectors along contact surfaces between solids. For this application, defining solids with exact elastic properties at each point P along the principal deformation direction (or "radial direction") defined by the normal vector $N(P)$ is sufficient.

To express exact non-linear elasticity in radial directions, we let $dR(Y)$ be a small radial force and dY the resulting small radial displacement. From equation (1) they must satisfy: $k_P(Y)dY = dR(Y)$. If we express deformations by variations in the field function, it yields:

$$df(Y) = Grad(f,Y).dY = Grad(f,Y).\frac{dR(Y)}{k_P(Y)} \quad (2)$$

As said previously, we want to use the way the field function varies inside the implicit solid to express physical properties. Let us directly model stiffness with the field's gradient:

$$\forall Y \quad Grad(f,Y) = -k_P(Y)\,N(Y) \quad (3)$$

This choice simplifies equation (2) which yields:

$$\int_{X_0}^{X(P)} df(Y) = -\int_{X_0}^{X(P)} (N(P).dR(Y)) = -N(P).R(P) \quad (4)$$

where the normal vector $N(P)$ remains constant during radial deformations. Let $g(P) = f(X(P)) - f(X_0)$ be the deformation field term associated at equilibrium with the radial force R. Here, the correspondence formula (4) becomes:

$$g(P) = -N(P).R(P) \quad (5)$$

We use equation (5) to define the general correspondence between deformations and forces characterizing implicit deformable solids. Used with a field function satisfying (3), this correspondence gives exact elastic properties in radial directions, but it associates no deformation at all with forces lying in the local tangent plane. Again, this is not a problem since the formula will only be used for computing the radial component of compression forces due to collisions.

Modeling stiffness with field functions

Let S be an object defined by a single skeleton and P_S a point of this skeleton. The field function along the segment between P_S and the closest point of the surface can be expressed as a function $f(r)$ of the distance $r(P) = d(P,P_S)$. From equation (3), the local stiffness at point P satisfies:
$k(P)\,N(P) = -f'(r(P))\;Grad(r,P)$.
But $Grad(r,P) = (P - P_S)/||P - P_S|| = N(P)$, so:

$$k(P) = -f'(r(P))$$

The resulting geometric representation of stiffness (the opposite of the field function's slope) facilitates the control of the simulated material. The user does not need to be a specialist in mathematical physics to easily design linear and non-linear elastic models as those of Figure 2. The field functions currently implemented are given in Appendix A.

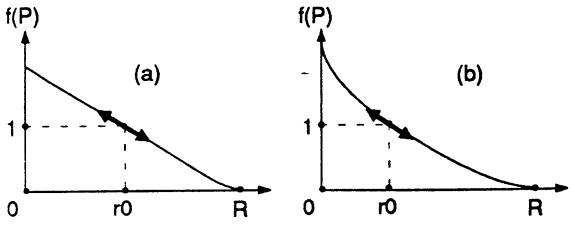


Figure 2: Examples of field functions.
(a) Linear elasticity: stiffness is constant during deformations.
(b) Non-linear elasticity: stiffness increases during compressions.

Homogeneity of the solids

When an object is generated by several skeletons, different field functions f_i can be associated with each one allowing non-homogeneous objects to be readily designed. The object behaves according to the local stiffness k_i in a zone influenced by a single skeleton. Otherwise, stiffness contributions from different skeletons blend together:

$$k(P)N(P) = -Grad(f,P) = -\sum Grad(f_i,P)$$

When "distance surfaces" are used, summing stiffness contributions in blending areas can be a problem. The stiffness may pass by a local extremum while varying between values associated with different skeletons. This problem corresponds to the bulge (or the narrowing) in shape which can appear when two fields superimpose [3]. Indeed, there is no reason why $||\sum_{j=1}^{n} Grad(f_j,P)||$ should take intermediate values between the k_i. In particular, homogeneous objects are not easy to model with distance surfaces. Giving the same field function to all the skeletons is far from sufficient.

"Convolution surfaces", for which field functions f_i are integrals of field contributions from each point of the associated skeleton, solve this problem. With this model, if the same field functions are used for several neighboring skeletons there is no bulge in shape nor in the stiffness function, so complex homogeneous objects can be designed. More generally, stiffness smoothly assumes intermediate values in areas influenced by multiple skeletons as does the field's gradient.

2.3 Animation of implicit deformable solids

Implicit deformable solids are especially suitable for a precise modeling of interactions. While penalty methods directly use the degree of interpenetration between objects to evaluate response forces, our model completely suppresses interpenetrations by introducing an intermediate "contact modeling" step between the detection and the response to collisions. The general animation algorithm is the following: At each time step,

1. Integrate the equations of motion for the rigid components of the solids by taking external forces F and torques T into account:

$$\sum F = mA$$
$$\sum T = I\dot{\Omega} + \Omega \wedge I\Omega$$

where I is the matrix of inertia of a solid computed from its rest shape. A represents linear acceleration and $\dot{\Omega}$ angular acceleration.

2. Displace flexible components from their rest shapes.
3. Treat interactions between objects:
   (a) Detect interpenetrations.
   (b) Model contact by deforming each solid in order to generate contact surfaces.
   (c) Integrate reaction and friction forces. Add them to the set of external actions to be applied to the rigid components at the next time step.
4. Display the objects with their new deformed shapes.

his algorithm is used with an adaptive time step. As ith penalty methods, overly deep interpenetrations genate overly large response forces (resulting, in the modelg contact phase, in excessive deformation of the objects). 'hen this situation is detected, the system recomputes the)jects positions using a smaller time interval.

The next section details the three steps of the interac-)n processing module and studies extensions to multiple llisions and to interactions with rigid implicit solids.

Interactions between Implicit Solids

1 Interpenetration detection

e use axis-parallel bounding boxes to quickly cull most n-intersecting cases. Afterwards, we benefit from the imcit representation of the objects, as in [10]. For each ir of solids, sample points associated with one of them e tested against the inside/outside function of the other. iis is done, of course, only for the sample points located iide the second solid's bounding box. As the list of solids eracting together is the only information needed for the)deling contact step, detection is stopped for a given pair solids as soon as an interpenetration point is found.

The method used for computing sample points at each ie step is detailed in Section 4. As will be shown, the tection process can be optimized by starting detection in e neighborhood of points (if any) that most penetrated e other object during the last time step.

2 Modeling contact

ce detected, an interpenetration must be suppressed by orming each object according to the set of interacting ids. Deforming objects involves generating contact surces as well as modeling the transverse propagation of demations (see Figure 3). Rather than simulating local inactions inside the objects, the system directly computes ormed shapes at equilibrium using a model for damped)pagation. Deformations outside a given "propagation a", an offset of the interpenetration zone, are ignored.

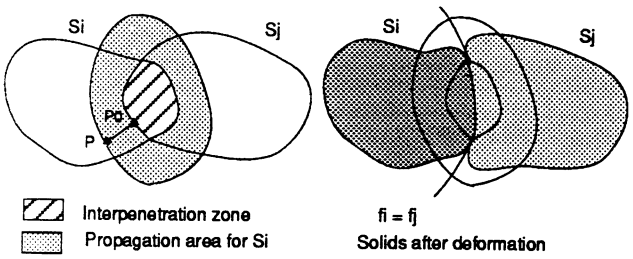


Figure 3: Modeling contact consists in applying different deformation fields in the interpenetration zone and in the "propagation area" associated with each solid (view in cross section).

Interpenetration areas: Generating contact surfaces

In addition to being a very natural model to express physical properties, the implicit surface formalism is also convenient for generating exact contact surfaces. See Figure 4.

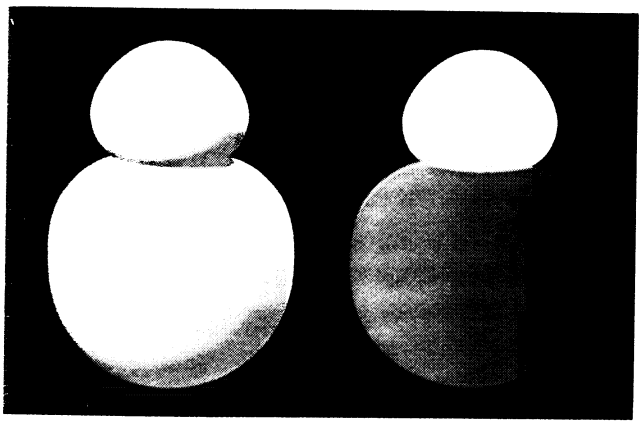


Figure 4: (left) Contact between two colliding objects. (right) View in cross section showing the exact contact modeling.

Suppose that two objects S_i and S_j interact locally. We are looking for new terms g_{ji} and g_{ij} to add to their respective field functions f_i and f_j in the interpenetration zone (g_{ji} represents the action of object j on object i). After deformation, the objects will be defined in this area by:

$$f_i(P) + g_{ji}(P) = 1 \quad (6)$$

$$f_j(P) + g_{ij}(P) = 1 \quad (7)$$

The deformation fields g_{ji} and g_{ij} must be negative (they model local compression of the objects) and locally generate a contact surface, thus equations (6) and (7) must have common solutions. In order to give the new contact surface exactly the same border as the interpenetration area, deformation fields must satisfy $g_{ij}(P) = g_{ji}(P) = 0$ in points P where $f_i(P) = f_j(P) = 1$. Moreover, the contact surface generated must fit with the local rigidities of colliding objects. So opposite forces must be applied by the two compressed objects on each point of this surface. Adding extra skeletons to generate deformation field terms would be inconvenient; Indeed, we prefer to directly use S_j's skeleton to deform S_i and vice versa. Consequently, the deformation field terms of

an interpenetration area are defined by:

$$g_{ji}(P) = 1 - f_j(P)$$
$$g_{ij}(P) = 1 - f_i(P)$$

With this choice, all the properties needed are verified: deformation field terms are negative in the interpenetration zone, and generate a contact surface defined by:

$$f_i(P) = f_j(P) \quad (8)$$

Let $P \in S_i$ be a point of the contact surface, $N_i(P)$ be S_i's unit normal vector, and $N_j(P) = -N_i(P)$.
Let $R_i(P) = ||R_i(P)||N_i(P)$ be the radial force applied by S_j at P. The correspondence (5) between forces and deformations yields : $g_{ji}(P) = -R_i(P).N_i(P) = -||R_i(P)||$, so: $R_i(P) = -g_{ji}(P)N_i(P)$. From equation (8), opposite forces are then applied by the objects along the contact surface:

$$R_i(P) = (1 - f_j(P))\ N_j(P) = (1 - f_i(P))\ N_j(P) = -R_i(P)$$

Deformations in "propagation areas"

We want to optimize the contact modeling process by directly computing deformed shapes in contact positions rather than simulating local interactions inside the flexible material. Designing a purely geometric layer is justified here: only deformations along contact surfaces will be used for computing response forces. The use of geometric propagation will not affect the motion at all. Moreover, we wish to model damped material where deformations outside given "propagation areas" can be neglected. Providing the user with a set of intuitive parameters, such as the thickness of the propagation areas around interpenetration zones or the way deformations are attenuated, offers a simple and efficient control of the simulated material.

More precisely, the user controls S_i's propagation field term $p_{ji}(P)$ (due to the collision with S_j) through two additional parameters in S_i's description:

- A thickness value w_i giving the size of the offset were deformations propagate around an interpenetration zone. Deformations will be neglected outside this area.
- An "attenuation value" α_i giving the ratio between the maximal value desired for p_{ji} and the current maximal compression term in the interpenetration area.

Because of the parameter α_i, the size of the bulge due to propagation of deformations will first increase during a collision, while the solid is progressively compressed, and then decrease back to zero when the colliding objects move off.

The propagation field p_{ji} must be positive within the propagation area in order to model a local expansion of the solid, compensating for the compression due to collision. To preserve the shape's first order continuity[2] p_{ji} and its derivative must become zero at the exterior limit of the propagation area, and have the same value and gradient vector as the contact term g_{ji} in the border of the interpenetration zone. Let P be a point within the propagation area, and P_0 the closest point of S_j in S_j's-gradient direction (see Figure 3). To satisfy the conditions just listed, we define p_{ji} along the line (P_0, P) by:

$$p_{ji}(P) = a_{k,a0,w_i}(d(P, P_0))$$

where $k = ||Grad(f_j, P_0)||$, a_0 is the maximal propagation value equal to α_i times the maximal compression field value, and $a_{k,a0,w}(x)$ is the piecewise polynomial function shown in Figure 5. An exact formula is given in Appendix B.

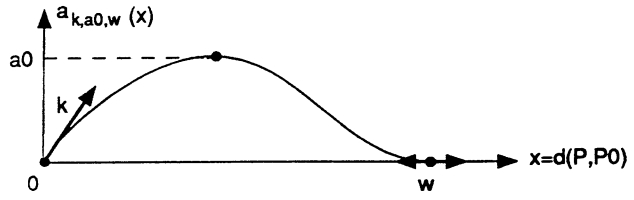


Figure 5: Attenuation function defining the propagation field.

With this choice, all the conditions on p_{ji} can be verified. Let us prove that $Grad(p_{ji}, P_0) = Grad(g_{ji}, P_0)$.

$$Grad(p_{ji}, P) = a'_{k,a0,w_i}(d(P, P_0))\ N_0$$

With the value of $a_{k,a0,w_i}$'s derivative in zero, we obtain:

$$Grad(p_{j,i}, P_0) = -Grad(f_j, P_0) = Grad(g_{ji}, P_0)$$

Expansion of the objects in propagation areas must not produce new interpenetrations. To best avoid this situation, we insure that each deformed object does not cross the median surfaces of equation $f_i(P) = f_j(P)$ (see Figure 3). In other words, $p_{j,i}(P)$ must be less than or equal to $1 - f_j(P)$ throughout the propagation area. If the problem does occur, the system truncates the propagation term and issues a warning that a smaller value should be chosen for α_i.

3.3 Computation of response forces

Radial reaction forces

The reaction forces directed along normal vectors are given by the correspondence (5) between forces and deformations. They are are numerically integrated along contact surfaces (this process is detailed in Section 4). Because of our choice for the contact surface, the principle that opposite reactions occur on two colliding objects is verified.

Friction and damping forces

To model both tangential friction in contact areas and damping due to the progressive compression of the solids, we include a friction coefficient λ_i in the description of each object. When a collision occurs, the friction and damping force F_i at a point P of the contact surface between S_i and S_j is expressed by:

$$F_i(P) = \lambda_i \lambda_j (V_j(P) - V_i(P)) \quad (9)$$

[2]The method would be easy to extend to higher order continuity by considering constraints over higher order derivatives, and by using more complex attenuation functions.

where $V_i(P)$ (respectively $V_j(P)$) is the speed of P, a point on the surface of the solid S_i (respectively S_j). Like radial reaction forces, friction forces are numerically integrated along contact surfaces.

Figure 6 shows the action of response forces during a few steps of an animation.

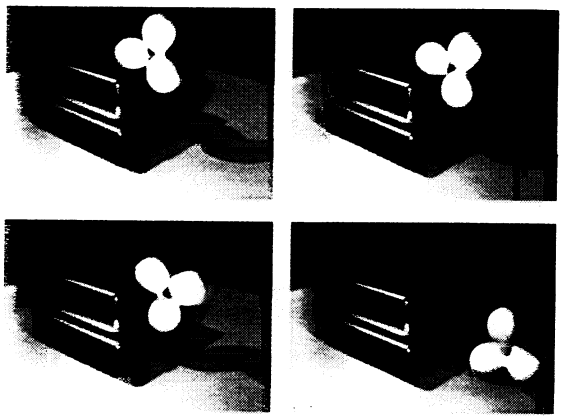


Figure 6: Flexible clover falling on a quite rigid staircase.

From collisions to lasting contacts

The deformed shapes generated during the contact modeling step can be conveniently used for lasting contacts and equilibrium states, because opposite forces are applied to each side of a contact surface. In Equation (9), F_i's tangential component represents friction due to the different tangential speeds of the solids at a contact point, while the normal component models the loss of energy due to the progressive deformation of the solids. The energy consumed over time enables colliding objects to settle into lasting contact situations, and then into resting stable states without unwanted oscillations. Figure 7 is an example of equilibrium state between four solids.

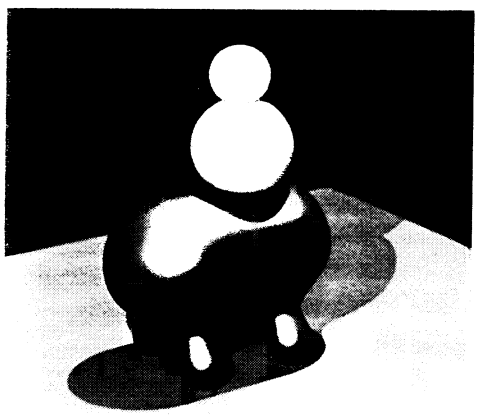


Figure 7: An equilibrium state between a rigid floor, a flexible vaulting horse, and two soft balls.

3.4 Multiple interactions

An important benefit of our model is that, in multiple interaction situations such as in Figure 7, the resulting shapes and reaction forces are *completely independent* of the order in which objects, or pairs of objects, are considered.

When an object interacts with several others, its compression field term (which produces the contact forces) is defined as a sum of terms due to the different collisions. No propagation term must be added in an interpenetration zone with another object, so, in practice, we always use a procedural method to compute field values. To evaluate the field generated by a deformed object S_0 at a given point P:

1. Compute the initial field value $f_0(P)$.
2. For each object S_i interacting with S_0, if P lies inside S_i, add the contact deformation term $1 - f_i(P)$.
3. If P was not lying inside any of the S_i, compute and sum all non-zero propagation terms at P. Truncate this sum if needed (as explained at the end of Section 3.2) before adding it to the field value.

If an intersection area is detected between more than two solids as in Figure 8, several negative compression terms are simultaneously added in this area. This leaves a small space between the solids, whose shapes remain C^1 continuous (generating multiple contact points would produce singularities).

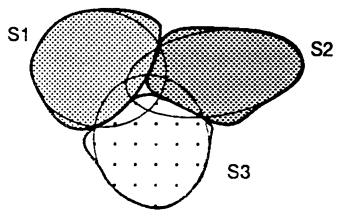


Figure 8: Deformation of 3 intersecting solids (cross sections).

3.5 Interactions with rigid implicit solids

Another important issue for our model is its ability to simulate interactions between flexible and rigid objects[3], like the vaulting horse and the floor in Figure 7.

Suppose an interpenetration has been detected between a rigid solid S_j and a flexible object S_i. The deformation field term applied to S_i in the contact area must make S_i exactly fit S_j's shape; i.e., the solutions of $f_i(P)+g_{ji}(P) = 1$ must be points satisfying $f_j(P) = 1$. Moreover, the deformation field term must be negative in the interpenetration zone given by $f_i(P) \geq 1, f_i(P) \geq 1$. Then, we define $g_{ji}(P)$ by:

$$g_{ji}(P) = (1 - f_j(P)) + (1 - f_i(P))$$

The usual formula is used for the attenuation function in S_i's propagation area. Simply, S_j's gradient vector (used to

[3] Analytical solutions such as those developed in [1] should be used for interactions between pairs of rigid objects.

define the slope of the attenuation function) is now replaced by $-(Grad(f_i + f_j, P_0))$ so that the bulge will exactly fit S_j's normal vectors at the border of the contact surface. When this is done, response forces corresponding to S_i's deformation are integrated along the contact surface, and opposite forces are applied to the rigid solid S_j according to the principle that opposite reactions occur on two colliding objects.

4 Implementation

Our modeling and animation system for implicit deformable solids is implemented in C++ on an SGI Indigo workstation. The current implementation uses distance surfaces which provide us with analytical expressions of normal vectors.

4.1 Optimizing the animation process

One of the main problems raised by implicit isosurfaces is the search for efficient ways to discretize objects. The animation process uses discretizations three times by animation step: for collision detection, for integrating response forces, and for displaying objects. Despite recent improvements in adaptive octree techniques, spatial partitioning polygonizations remain quite expensive. Using this type of algorithm at each time step would prevent any interactive computation and display of the animation.

Fortunately, the solids only deform locally during animations and return to their rest shapes. Their topology never changes (otherwise, our hybrid model with its invariant matrix of inertia would be invalid). Consequently, the objects need not be completely re-sampled at each time step.

Before an animation is calculated, sample points and the associated normal vectors are precomputed form the object's rest shape, and stored relative to the local coordinate system. Then, at each animation step:

- The sample points are positioned according to the current position and orientation of the solid's rigid component.
- They are used to detect collisions. To benefit from temporal coherence, tests for interpenetration are first performed in the neighborhood of points P_j which penetrated most deeply into the other object at the previous time step[4].
- Finally, the sample points in deformed areas are recomputed, before display, by using a linear search algorithm along the undeformed normal direction. The use of this direction insures that the points will come back to their initial positions after any deformation.

  To improve in efficiency, response forces in contact areas are integrated during this process. If a point is located in an interpenetration zone, the field function computes the deformation term, which is equal to the local reaction force. As soon as a point of the contact surface is found, this force (plus the friction force term), multiplied by the area of an elementary surface ds, is added to the sum of external actions applied to the object. In the current implementation, ds is approximated by an average value computed from the size of the discretization voxels.

[4] We use a continuation method for sampling, so starting detection a few points before P_j in the list of sample points is sufficient.

4.2 Rendering

The implicit formalism provides us with exact high level descriptions of deformed solids, even if only a few sample points are used during the computations. During animations, we directly save the parameters needed to compute the deformed field functions defining the objects (including stiffness, scope of influence, attenuation parameters, and the current list of colliding objects). To give an idea of required disc space, the file describing the implicit objects of Figure 7 take less than 1 Kbyte, while the storage of the associated sample points with their normal vectors and the list of triangles takes more than 1200 Kbyte.

Once the objects are stored, any method could be used for rendering, including computing polygonizations with an arbitrary precision. We currently use direct ray-tracing on implicit surfaces, implemented as an extension to the public domain renderer *Rayshade* (by C. Kolb). If a ray intersects an implicit solid's bounding box, we first look for a seed point along the ray which is located inside the solid. If we find one, an intersection point is computed by binary search. Testing if a point is inside or outside is done by evaluating the solid's potential field. Normal vectors at the intersection points are analytically computed from the field gradient.

5 Conclusion

This paper presents a novel way to model deformable solids. The implicit formalism provides a compact and unified formulation for both geometric and physical properties, and enables to keep a continuous high level representation of the objects. The model offers simple and quite general control of the simulated material. One can experiment with field function curves to adjust stiffness variations when objects are compressed, or with different ways to propagate deformations due to collisions. All the parameters are easy to understand, even for a non-specialist.

Well adapted to local deformations, the system is especially designed for a precise modeling of interactions. It generates exact contact surfaces between solids which facilitates a precise evaluation of reaction forces. The model applies to sudden collisions, lasting contacts, and equilibrium situations. It gives an elegant solution to the multiple collision problem, producing new deformed shapes and response forces which are independent of the order in which collisions are detected. The model can be generalized to treat interactions between flexible and rigid objects.

During animations, discretized representations of the objects are displayed at interactive rates, while a compact storage of their implicit description is performed. This description, which still defines curved contact surfaces, is used for

roducing subsequent, high-quality images.

uture work

mplementing convolution surfaces [3] would facilitate the esign of complex objects composed of homogeneous materials. To improve generality, these surfaces should be extended to non-exponential potential fields.

At present, deformed shapes only depend on the set of xternal forces currently applied to the solids, so deformaons disappear as soon as there is no longer contact between bjects. Modeling visco-elastic or elasto-plastic behaviors in ddition to pure elasticity would be a good extension. Morever, some objects of the real world conserve their volume uring deformations while others are partially compressible. lodeling the propagation of deformations according to a mpressibility parameter would provide easier control.

The implicit deformable model opens new directions for ture research, particularly within the area of human simation. Modeling complex objects such as deformable flesh vering rigid skeletons could not be done with previous obal deformation techniques (the skeleton must not be drmed, nor the flesh on the opposite side of the skeleton). odal approaches can be used, but they demand complited databases [6] and involve an expensive numerical simution of deformations propagating in damped material. Our ility to model local deformations while preserving a comct continuous representation of objects, even when they e non-homogeneous, would be helpful. Moreover, the pree contact modeling presented here should allow human odels to interact with the simulated world.

cknowledgements

ny thanks to Jean-Dominique Gascuel for his constant supt during this research and for implementing direct ray-tracing implicit surfaces. Thanks to Philippe Limantour, Christophe del, Frédéric Asensio and Julien Signes for interesting early cussions, and to François Sillion, Alain Chesnais, Dave Forsey, il Brock and Jules Bloomenthal for re-reading this paper.

eferences

[1] David Baraff. Dynamic simulation of non-penetrating rigid bodies. *PHD Thesis*, Cornell University, May 1992.

[2] David Baraff and Andrew Witkin. Dynamic simulation of non-penetrating flexible bodies. *Computer Graphics*, 26(2):303–308, July 1992. Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 1992).

[3] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

[4] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, March 1990.

[5] Marie-Paule Gascuel, Anne Verroust, and Claude Puech. A modeling system for complex deformable bodies suited to animation and collision processing. *Journal of Visualization and Computer Animation*, 2(3), August 1991. A shorter version of this paper appeared in *Graphics Interface'91*.

[6] Jean-Paul Gourret, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of object and human skin deformations in a grasping task. *Computer Graphics*, 23(3):21–29, July 1989. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989).

[7] Annie Luciani, Stéphane Jimenez, Olivier Raoult, Claude Cadoz, and Jean-Loup Florens. An unified view of multitude behaviour, flexibility, plasticity, and fractures: balls, bubbles and agglomerates. In *IFIP WG 5.10 Working Conference*, Tokyo, Japan, April 1991.

[8] Gavin Miller. The motion dynamics of snakes and worms. *Computer Graphics*, 22(4):169–177, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[9] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4):289–298, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[10] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222, July 1989. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989).

[11] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3-D models with local and global deformations : deformable super quadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(7):703–714, July 1991.

[12] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

[13] Demetri Terzopoulos and Andrew Witkin. Physically based model with rigid and deformable components. *IEEE Computer Graphics and Applications*, pages 41–51, December 1988.

[14] Andrew Witkin and William Welch. Fast animation and control for non-rigid structures. *Computer Graphics*, 24(4):243–252, August 1990. Proceedings of SIGGRAPH'90 (Dallas, Texas, August 1990).

Appendix A. Equation for the field functions

Field functions currently implemented are parameterized by a scope of influence R, a thickness value r_0, and a stiffness value k:

$$\begin{aligned} f_i(P) &= ar^2 + br + c && \text{if } r \in [0, r_0] \\ f_i(P) &= (r-R)^2(dr+e) && \text{if } r \in [r_0, R] \\ f_i(P) &= 0 && \text{elsewhere} \end{aligned}$$

$d = -(k(r_0-R)+2)/(r_0-R)^3$, $e = (kr_0(r_0-R)+3r_0-R)/(r_0-R)^3$.

If linear elasticity is chosen, $a = 0$, $b = -k$ and $c = kr_0 + 1$. If non-linear elasticity is selected, we use $a = k/(2r_0)$, $b = -2k$, $c = 3kr_0/2 + 1$ (of course any other stiffness variation inside the object would be easy to implement).

Appendix B. Equation for the attenuation function

The equation we use for the attenuation function is:

$$\begin{aligned} a_{k,a0,w}(r) &= cr^3 + dr^2 + kr && \text{if } r \in [0, w/2] \\ a_{k,a0,w}(r) &= \frac{4a0 + (x-w)^2(4x-w)}{w^3} && \text{if } r \in [w/2, w] \end{aligned}$$

Where $c = 4(wk - 4a0)/w^3$ and $d = 4(3a0 - wk)/w^2$.

Chapitre 4

Highly deformable material

Ce travail a fait l'objet d'une présentation à un colloque international, ainsi qu'à un atelier français :

[**DG94**] Highly Deformable Material for Animation and Collision Processing. Mathieu Desbrun et Marie-Paule Gascuel. In *5th Eurographics Workshop on Animation and Simulation*, Oslo, Norvège, septembre 1994.

[**DG94F**] Simulation de matière hautement déformable. Mathieu Desbrun et Marie-Paule Gascuel. Dans les actes du colloque *Les Simulateurs* organisé par le "Groupe de Travail Français Animation et Simulation", Lille, octobre 1994.

Résumé

Cet article présente une méthode pour modéliser et animer des matériaux "hautement déformables" comme de la pâte à modeler, de la glaise, ou de la boue semi-liquide. Ces matériaux sont inélastiques : ils absorbent une partie des déformations qu'ils subissent, et reviennent rarement à leur forme initiale. Leurs surfaces sont généralement lisses et épousent la forme des objets avec lesquels ils sont en contact. De plus, la topologie de ces corps est variable : ils peuvent se fracturer en plusieurs parties au cours d'une animation.

Nous proposons ici un modèle hybride. Chaque objet est constitué d'un composant inélasique discret, enrobé d'une couche de matériau déformable. Cette dernière est définie grâce à une formulation implicite, convenant particulièrement bien à la modélisation d'objets dont la tolopogie change au cours du temps. Bien adaptée à la détection et au traitement des collisions, la couche déformable engendre et maintient de véritables surfaces de contact entre objets en interaction. Les forces de réaction intégrées le long de ces surfaces sont transmises à la structure interne, qui se déforme en conséquence. La simulation de l'ensemble est interactive.

Highly Deformable Material for Animation and Collision Processing

Mathieu Desbrun
Marie-Paule Gascuel

iMAGIS / IMAG[1] - INRIA
BP 53, F-38041 Grenoble cedex 09, France

email: Mathieu.Desbrun@imag.fr

Abstract

This paper presents a method for modeling and animating highly deformable material such as clay, dough or mud. The animated objects are inelastic: they absorb deformations and seldom come back to their initial shape. Their surfaces are smooth and fit with the objects they are in contact with. Moreover, their topology may change during animation sequences. In particular, they can break into pieces.

The new model presented here is an hybrid one. Each object is composed of an elastic coating over a discrete inelastic layer. The elastic material is defined by an implicit formulation that is particularly convenient for modeling objects whose topology changes over time. Well-suited to automatic collision detection and response, the elastic layer generates and maintains exact contact surfaces between interacting objects. Response forces computed along contact surfaces are transmitted to the internal structure that deforms accordingly. The whole simulation is produced at interactive rates.

Keywords: animation, simulation, deformation, soft material, implicit surface, adaptive sampling, collision detection, collision response.

1 Introduction

In traditional animation systems, specifying motion and successive shapes of deformable material interacting with the simulated world requires a great amount of specialized knowledge from the animator. Recently, several physically-based models have been proposed for the automatic computation of motion and deformations. Most of them are restricted to elastic material, which comes back to its rest shape after any deformation.

Highly deformable objects that can simulate various behaviors from quasi-solid to quasi-liquid, such as those made of mud, dough or clay, are difficult to animate with current tools. In particular, maintaining well-defined and smooth surfaces on objects whose topology may change over time remains a challenge.

[1]IMAG is a Research institute affiliated with CNRS, Institut National Polytechnique de Grenoble, and Université Joseph Fourier.

This paper presents a new method combining deformations of surface and topologic changes. Based on an implicit formulation, our model maintains exact contact surfaces between inelastic objects.

Previous Approaches

A few models for inelastic deformations have been developed in Computer Graphics. All of them belong to the class of nodal approaches: deformations are expressed by displacement of elementary nodes representing mass points or sample points inside the material.

Some of the approaches are derived from elastic models based on discretized continuous equations that are extended for modeling inelastic deformations. A layered model [TF88] represents inelastic material as an elastic component at rest with respect to a reference component used for computing motion. Inelasticity is modeled by allowing the reference component to progressively absorb some of the deformations of the elastic layer. Here, topology changes are limited. Fractures can be modeled, but only at predefined fracture points. Another system [TPF89] models blocks of flexible material that can be heated until they melt. A lattice of mass points linked by springs of variable stiffnesses is used at the solid state, while particle interaction laws govern motion of mass points that have been disconnected from the network (each spring disappears at a certain heat). The main drawback of these hybrid models is the non-unified view of the different behaviors. Specific laws are developed for each behavior to model, but there is no general method. For instance, freezing the material back to a solid state would be quite difficult using the last approach.

Another solution is the direct use of discrete elementary masses, each of them being animated separately according to a set of interaction laws with the others [LJF+91]. These physically-based "particle systems" give an unified view of multiple behaviors, from elasticity to plasticity and from quite stiff objects to quasi-liquid states [Ton91, LJR+91]. In addition, fractures and other tolopogical changes can be generated very easily.

Methods for collision detection and response have to be provided together with a deformable model. Detecting interpenetrations between objects represented by lattices of elementary nodes basically requires $O(n^2)$ steps, where n is the number of nodes. Response to collisions between deformable solids and rigid objects is often computed from penalty methods [TF88, TPF89, Ton91]. A force proportional to the amount of interpenetration is applied to the nodes that have penetrated a rigid object. When all the objects of the scene are represented by sets of elementary masses [LJF+91, LJR+91], interactions between objects are just a particular kind of interaction between particles: a repulsion force is produced between two mass points from different objects that become too close (otherwise, this force is zero). Computing interaction forces between n mass points from different objects normally takes $O(n^2)$ steps, but the use of adequate data structures can reduce this complexity to $O(n \log(n))$ [Hou92].

None of these models generates any contact surfaces between highly deformable objects. This is mainly due to the fact there is no surface defined. An object is just a set of nodes or of mass points, and there is no information

telling which points define "the object surface" after it undergoes topological changes. However, displaying objects at a macroscopic level rather than just as a set of points is often needed. Some of the authors display implicit surfaces generated by the mass points [TPF89, Ton91]. But as these surfaces are not considered for collision detection, this can result in visual anomalies such as local interpenetrations and bouncing before contact.

Overview

This paper presents a new model for highly deformable material that provides exact surface contact during interactions. We use an hybrid representation enabling a coherent use of internal and external structures. Each object is composed of discrete internal elements linked by interaction laws, and coated with some elastic material. Based on an implicit formulation particularly convenient for modeling topological changes, the elastic flesh defines a soft surface for each block of material. This surface is used for detecting collisions (which can be done efficiently thanks to the implicit inside-outside functions), computing deformed shapes, and integrating response forces. These forces are then propagated into the internal structure, producing inelastic deformations and possibly fractures, in the subsequent time steps.

Section 2 briefly reviews the model for implicit elastic material introduced in [Gas93]. Section 3 explains how to extend this model for defining an elastic coating around a discrete layer. In particular, volume variations of the implicit flesh must be carefully controlled during a progressive compression of the internal structure. This leads to coherent choices for the internal and external parameters. Section 4 details our method for collision detection and response for the hybrid model. Computing sample points for collision processing is a difficult task, since the topology of the objects changes over time. Therefore, we propose an efficient adaptive sampling algorithm. Section 5 presents our results and discusses work in progress.

2 Implicit Elastic Material

This section briefly describes the continuous elastic model developed in [Gas93]. Specifically designed for precise contact processing, this model uses isopotential implicit surfaces generated by "skeletons" to model both the geometry and the physical properties of the objects.

Implicit Surfaces

An implicit surface S generated by a set of skeletons $S_i (i = 1..n)$ with associated "field functions" f_i is defined by: $S = \{P \in I\!R^3 \ / \ f(P) = 1\}$ where $f(P) = \sum_{i=1}^{n} f_i(P)$ [2]. The skeletons S_i can be any geometric primitive admitting a well defined distance function. The field functions f_i are decreasing functions of the

[2] Throughout this paper, we always use the value 1 as the "isovalue" defining the isosurface of an object.

distance to the associated skeleton (we call e the distance for which the field is 1, k the slope in e, and R the radius of influence of the skeleton; see Figure 1). The surface surrounds a volume defined by $f(P) \geq 1$, and normal vectors $N(P)$ are directed along the gradient of f.

Figure 1: Field generated by two points, and the resulting implicit surface.

Modeling Elasticity with Field Functions

The set of points verifying $f(P) = 1$ being fixed, the variation of f around the isosurface can be used to model physical properties. Therefore, stiffness can be directly stored in the field's gradient by choosing f such as: $\vec{\nabla} f(Y) = -k_P(Y)\,\vec{N}(Y)$, where $k_P(Y)$ is stiffness at point P when this point has been moved to position Y. Elasticity can then be expressed in terms of field functions:

$$df(Y) = \vec{\nabla} f(Y) \,.\, d\vec{Y} = \vec{\nabla} f(Y) \,.\, \frac{d\vec{R}(Y)}{k_P(Y)} = \vec{N}(P) \,.\, d\vec{R}(Y)$$

If the small applied force $d\vec{R}$ is radial, the normal vector remains unchanged during deformation, and integrating the last equation yields a very simple correspondence between applied force and deformation : $g(P) = \vec{N}(P) \,.\, \vec{R}(P)$, where $g(P) = \int df(Y)$ is the deformation field term. This correspondence will be used during animations for computing radial response forces from deformations of colliding objects.

Collision and Contact Processing

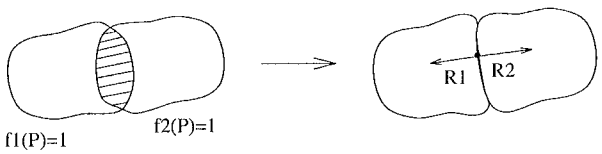


Figure 2: Deformation and response during a collision.

In [Gas93], implicit elastic material is attached to a rigid component used for computing motion (variations of the inertia tensor due to deformations are neglected). The implicit layer processes collisions and deformations, and provides the rigid component with response forces to be applied at the next time step. Collision processing is performed in three steps (see Figure 2) :

1. Detect interpenetrations between objects. After a predetection using bounding boxes, use the implicit inside/outside functions by testing one solid's sample points against the other's field function.

2. Generate exact contact surfaces by deforming the solids. Apply a negative compression field $g(P)$ in the interpenetration area, and a positive dilatation field in the propagation zone in order to efficiently model the transversal propagation of deformations in damped material.

3. Compute the response forces and the torques to be applied to the rigid components at the next time step. Our method for generating exact contact surfaces takes the objects' local stiffnesses into account, so opposite compression forces $\vec{R}(P) = g(P)\vec{N}(P)$ are applied to both sides of the contact surfaces. These forces are integrated together with fluid friction forces.

3 A Hybrid Model for Highly Deformable Material

As emphasized in the introduction, a simple and unified way of modeling from stiff to very soft behaviors and for simulating material capable of breaking into pieces is to use physically-based particle systems. However, the main drawback of these systems is the lack of methods for defining a soft surface for the objects, which could be used for collision processing and for display.

The main point of our approach is to adapt the elastic material presented in Section 2 to an implicit coating around a generalized particle system. In other words, each particle interacting with the others through long range attraction forces and short range repulsion forces will be seen as a geometric skeleton contributing to the implicit surface of an object. During an animation, the relative motions of the skeletons will automatically produce adequate deformations and topological changes of the objects' surfaces. An important benefit of the elastic implicit coating will be to provide a good model for detecting and processing collisions.

Animation of the Internal Structure

As explained previously, skeletons defining an implicit surface may be any geometric primitive admitting a well defined distance-function (points, segments, curves, triangles, simple volumes..). To model a block of highly deformable material, we choose the number of skeletons, their shapes, and the flesh thickness around each skeleton according to the shape and size of the macroscopic components that we want to be unbreakable during any animation. Our approximation consists in considering, while computing motion, that the mass of the implicit body is entirely concentrated in the skeletons. Each skeleton is represented as a rigid body, with its associated mass and inertia tensor, animated by integrating over time rigid body equations of motion (or point-wise dynamics if the skeleton is a point). Skeletons are linked together by adequate interaction forces, constituting a generalized physically based particle system.

We use Lennard-Jones interaction forces as in [Ton91, Jim93]. The force to be applied to the center of a skeleton S_1 interacting with another S_2 is :

$$\vec{F} = K \left(\frac{r_0^n}{r^{n+1}} - \frac{r_0^m}{r^{m+1}} \right) \frac{\vec{r}}{r} \qquad (1)$$

where $\vec{r} = S_1 - S_2$ and $r = ||\vec{r}||$ The remaining parameters of Equation (1) can be used as follows: r_0 represents the rest distance between the skeletons, K controls the softness of the system (when K increases, the global behavior is stiffer). The assumption $n = 2m$ is commonly used. We are currently experimenting highly deformable material with values $m = 4, n = 8$.

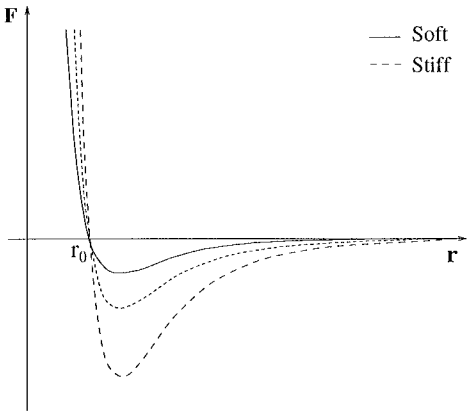


Figure 3: Interaction laws for three different values of K.

The use of these interaction forces is not sufficient for modeling viscous behaviour as particles may oscillate indefinitely around a rest position. We avoid this problem by adding damping forces to the model. The damping force applied to a particle should depend on the local density of particles around it. To achieve this, for each particle we compute a damping force with respect to every other particle that interacts with it. Damping in a viscous medium is usually proportional to the square of the object speed. Here, only the relative speed of the two interacting skeletons will produce damping. The total damping force applied on a skeleton S of speed vector $\vec{V}$, and interacting with the skeletons S_i of speed vectors $\vec{V_i}$, is given by:

$$\vec{D} = \sum_i \delta_i(\text{dist}(S, S_i)) \, ||\vec{V_i} - \vec{V}|| \, (\vec{V_i} - \vec{V})$$

where δ_i is a damping coefficient based on the distance between S and S_i. In practice we use a simple function with only two parameters: r_1 controlling the area of influence and α providing the maximum influence, i.e.:

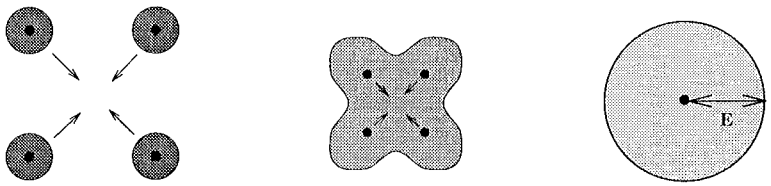
$$\delta_i(r) = \begin{cases} \alpha.\frac{(r-r_1)^2}{r_1^2} & \text{if } r \in [0, r_1] \\ 0 & \text{otherwise} \end{cases}$$

These damping forces offer easy tuning of the viscosity of the deformable materials.

Coherent Volume Variations

Simply animating the skeletons of an implicit solid while using an arbitrary field function to coat them would not produce good results. If no particular

care is taken in the choice of parameters, applying compression forces to internal structures could dramatically increase the volume. Figure 4 shows what would happen if we model highly deformable material with a quadratic field function as used in [Gas93], where all skeletons were motionless in the object local coordinate system.



(a) (b) (c)

Figure 4: Compression of the internal layer should not increase the volume!

Studies of materials that conserve their volume during deformations has already been performed in Computer Graphics [PB88]. To model more general behaviors, a certain amount of volume variation can be allowed. Compression forces applied to the internal structure of the material in Figure 4 should produce a compression of the total volume, but never a dilatation.

Wyvill *et al.* [WMW86] previously pointed out the fact that two skeletons in the same place should create a sphere with twice the volume of a one-skeleton sphere. The choice of their *isovalue* ensures this constraint only when two point-skeletons merge. Let us express this problem more generally, when any number of particles are concentrating in one point (see Figure 4). Suppose that n punctual skeletons are initially in positions where their field functions do not blend together, and that they move to the same location. Initially composed of n spheres of radius $e = f^{-1}(1)$, the object blends into a single larger sphere of radius $E = (n.f)^{-1}(1) = (f)^{-1}(1/n)$. We write:

$$\frac{4}{3}\pi E^3 \leq n \cdot \frac{4}{3}\pi e^3 \quad \text{which is equivalent to:} \quad f^{-1}(\frac{1}{n}) \leq \sqrt[3]{n} \cdot e$$

so that the global volume will decrease during this process. This last equation must be verified for every n smaller than the number of particles in our system: this gives us a set of constraints for f. Combining the last equation with the fact that f is a decreasing function and taking $x = \sqrt[3]{n}.e$ yields:

$$\forall x \geq e \quad f(x) \leq \left(\frac{e}{x}\right)^3 .$$

In particular, the continous function $f(x) = (e/x)^3$ verifies all the constraints, and so maintains volume between the configurations of Figure 4(a) and Figure 4(c).

However, even if the volume is constant or decreases between the two extreme configurations of Figure 4, a maximum may appear in an intermediate position. Let us consider an implicit solid generated by two punctual skeletons. Figure 5 represents the volume[3] variations as a function of the distance between

[3] As the formula expressing volume cannot be integrated analytically, we have approximated volume by discretizing space into small voxels.

the skeletons. For the field $f(x) = (e/x)^3$, there is a maximum near $d = 2e$, where e is the radius of the implicit spheres associated with each skeleton, which represents a 19% increase of the volume value. This result is still better than the 43% increase produced by the quadratic field of [Gas93]. Moreover, even if a field function admits a maximum value for volume, coherent choices for the internal and external parameters of our hybrid model can produce correct behaviors. In Figure 5, if the distance between the skeletons oscillates around the equilibrium position $d = e$, the volume will, as expected, decrease during a compression and increase during a small dilatation of the internal structure. Except during fractures, distances between skeletons in a block of deformable material do not vary much during a simulation: the block reaches an equilibrium state, and skeletons oscillate near rest positions. So in practice, we choose in Equation (1) the rest distance r_0 between the skeletons to be equal to the middle value e of the "validity interval" for which coherent compressions and dilatations are produced.

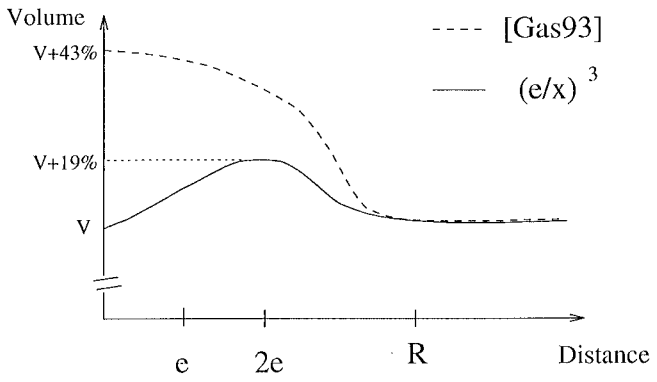


Figure 5: Volume as a function of the distance between two skeletons.

4 Animation and Collision Processing

The general animation algorithm for highly deformable material can be described as follows. At each time step:

1. Compute new positions for the skeletons, by integrating the associated equations of motion from the set of externally applied forces.

2. Detect collisions, avoid interpenetrations by generating exact contact surfaces between objects, and compute response forces.

3. Display the resulting deformed shapes.

The first phase requires smaller time steps than the one used for display, so deformations and response forces are calculated every Δt, whereas integration of particles movement are performed every $dt = \Delta t/k$.

Our method for collision detection and response is similar to the algorithm given in Section 2. However, as objects may break into pieces during simulations, bounding boxes can become too large, i.e. inefficient, and finding sample

points at each time step becomes a more serious challenge. Moreover, performing these operations very efficiently is essential for computing the animation at interactive rates. Another problem that is introduced when modeling highly deformable material is defining how response forces should be transmitted into the internal structure of the solids. The next two sections address these problems.

Efficient Adaptive Sampling for Topology-Variable Objects

Contrary to elastic implicit solids that only deform locally and then recover their initial shape, blocks of highly deformable material may suffer any topological change during a simulation. A violent collision may create a hole, a block can break into pieces, and several blocks made of the same material can melt[4]. In consequence, pre-sampling the objects before a simulation and then always using the same sample-points, as was done in [Gas93], is no longer possible. Unfortunately, spatial partitioning polygonization is compute-intensive, thus using these techniques at each time step would prevent any interactive computation and display of the animation.

Consequently, we have developed a new approach for an efficient adaptive sampling of highly deformable material. Based on the idea of seed migration introduced in [BW90], our method benefits from temporal coherence between the frames of an animation. The basics are the following: each skeleton sends a set of seed points to sample the surface, with a good spatial distribution around it. A seed, always moving along a fixed axis with respect to its skeleton's local coordinate system, can be efficiently recomputed at each time step from its last position in this coordinate system : very few search steps are required (because of time coherence) to find an inner and an outer point to start a binary search on the field function (see Figure 6).

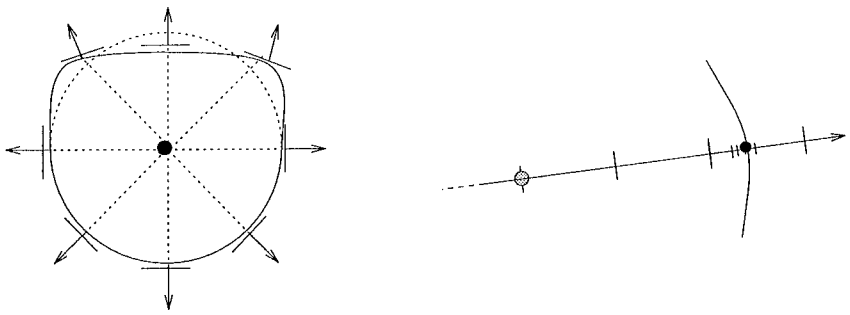


Figure 6: Seed migration during deformation by large steps and binary search.

To increase efficiency and to ensure a reasonable distribution of seed points when several basic volumes melt together, some of the seeds are invalidated at each time step (see Figure 7): during search steps, if a seed enters an area where the field generated by its own skeleton is smaller than another field contribution,

[4] In our current implementation, the user defines one or several blocks of highly deformable material, each one being breakable into several pieces. Collisions are computed between pieces from different materials while components from the same object always "melt" together.

the seed stops and is marked as invalid. An invalid seed may reappear if the topology of the object changes.

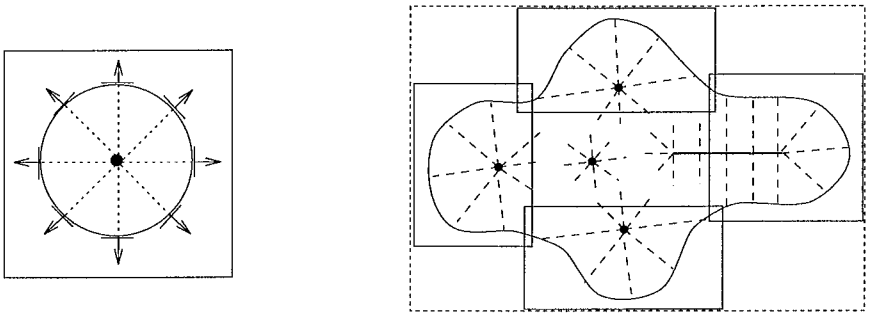


Figure 7: Seed points used for sampling and associated bounding boxes.

Another problem is defining bounding boxes for optimizing collision detection between highly deformable objects. Storing an axis-parallel bounding box in each object's local coordinate system [Gas93] cannot be done, since an object may be composed of several pieces and each skeleton comprising an object has its own local coordinate system. Moreover, using this large box would be inefficient when an object has broken into pieces. Therefore we attach an axis-parallel bounding box to each skeleton. The size of the box is computed from the position of the valid seeds sent by this skeleton so that the object surface – but not necessarily the object volume – is included in the union of skeleton-boxes. Then a large bounding box is computed from all those boxes as shown in Figure 7. Collision detection uses this hierarchy of boxes to reduce the number of field evaluations. If two large boxes from different objects interpenetrate, collisions are detected between pairs of skeleton-boxes. For each colliding pair, only the seed points attached to one box and located inside the other are tested against the other object's field function. A useful extension would be to add to the hierarchy an intermediate box computed from each connected component of an object.

Response to Collisions: Transmitting Forces to the Internal Structure

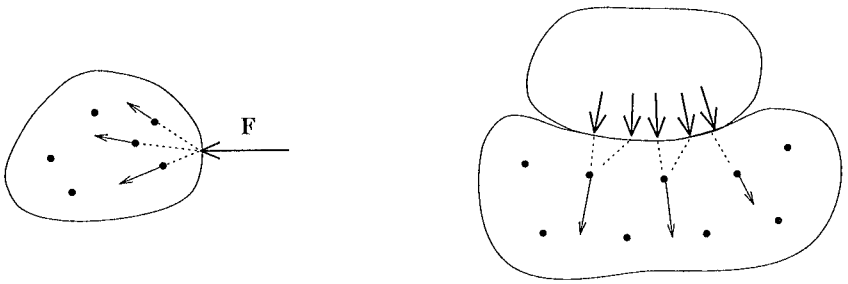
After the "modeling-contact" step, which works in exactly the same way as in Section 2 (deformation fields generating exact contact surfaces are applied to colliding objects), response forces computed on contact surfaces must be transmitted to the internal structure. However, in contrast to the method in Section 2, there is no-longer one rigid component to which send the forces, but several different skeletons, each contributing to the field value.

If an external force is applied at a point P of the surface (that verifies $\sum_i f_i(P) = 1$) a first approach consists of distributing this force between the skeletons S_i in proportion to their amount of contribution to the normal vector at P. See Figure 8(a). A good way to do this, especially if the applied force is directed along the normal vector at P, is to use the decomposition of the

normal vector at P into the sum of gradient vectors from each skeleton :

$$\vec{N}(P) = \frac{\sum_i \vec{\nabla} f_i(P)}{||\sum_i \vec{\nabla} f_i(P)||}$$

So, if we let $\vec{n_j}(P)$ be: $\vec{n_j}(P) = \vec{\nabla} f_j(P)/||\sum_i \vec{\nabla} f_i(P)||$, a radial force applied on the initial shape $\vec{F} = F\vec{N}(P)$ at P can be split into forces $F\vec{n_j}(P)$ applied to each skeleton S_j, together with the associated torques if needed.



(a) (b)

Figure 8: Two different solutions for transmitting forces to the skeletons.

This first method gives a good distribution of punctual forces applied on an object surface without collision, but this is too restrictive. However, in the case of response forces occurring across an entire contact surface after collisions, the algorithm can still be simplified. Our seed points are, in effect, a precise sampling of a smooth contact surface between two deformable objects. So, transmitting the force at a seed point to the skeleton that contributes the most to the potential at this point is quite sufficient. Moreover the invalidation test seen in section 4 eases the computation, because it ensures that the response force calculated on a seed will be transmitted to the skeleton that sent this seed. So a skeleton will be affected proportionally to its contribution to the contact surface sampling. See Figure 8(b). The interactions inside the system of skeletons will propagate the action of the external forces throughout the structure. The simplified algorithm produces good results, as demonstrated by the animations that we describe in the next section.

5 Results and Concluding Remarks

Animation Sequences

The sequence depicted in Figure 8 has been computed with only 18 point-skeletons for the block of material. This sequence shows the result of weak interaction forces that create a viscous fluid behavior. The second one uses stronger interaction forces, preventing objects from breaking up on impact. These objects contains only 12 and 8 particles respectively (see Figure 9). The benefit of detecting collisions with the exact surface is that this surface does not need to stay very close to the particles. This enables us to generate very smooth objects with few skeletons.

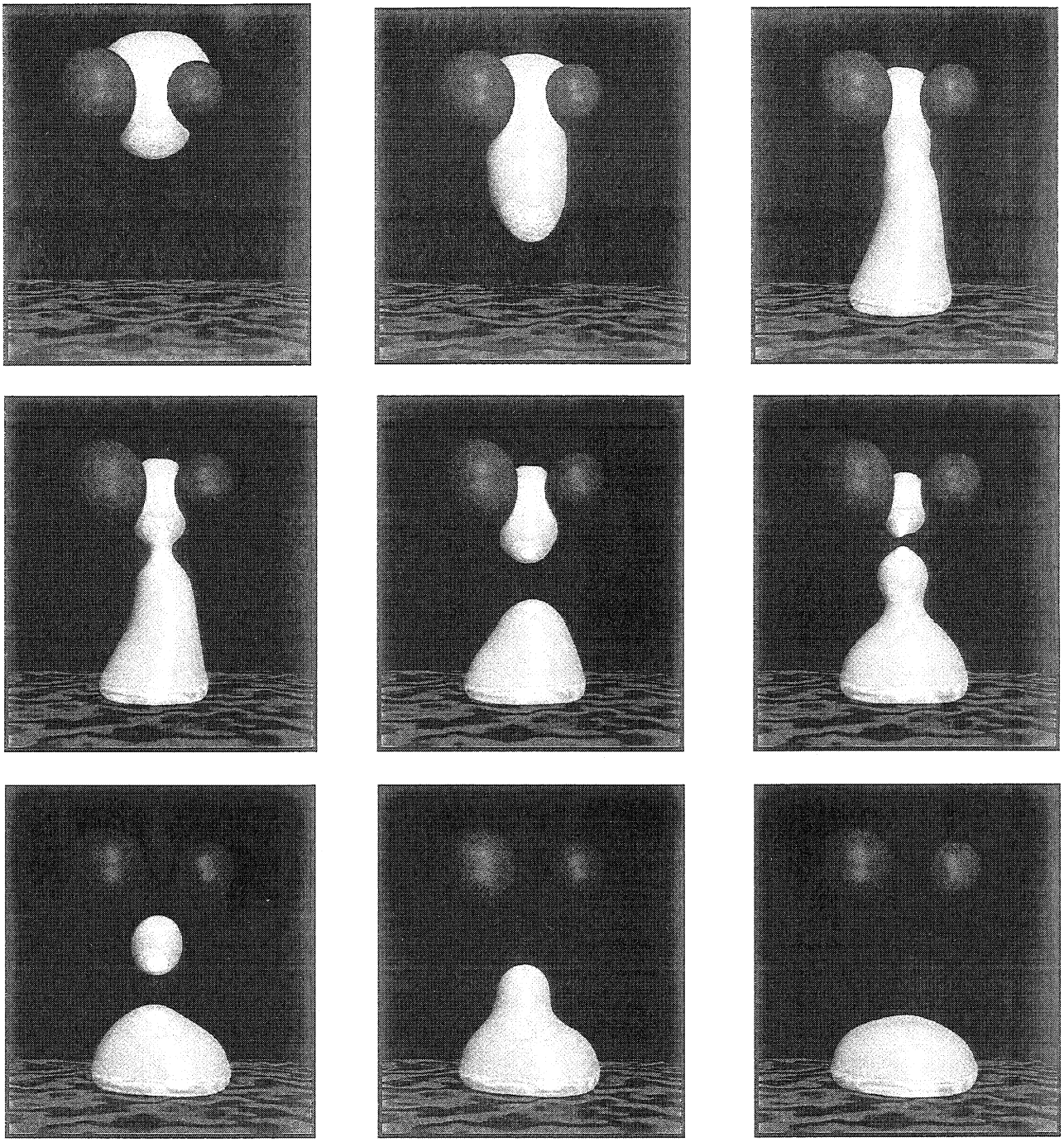


Figure 8 : Highly deformable substance falling between two rigid obstacles.

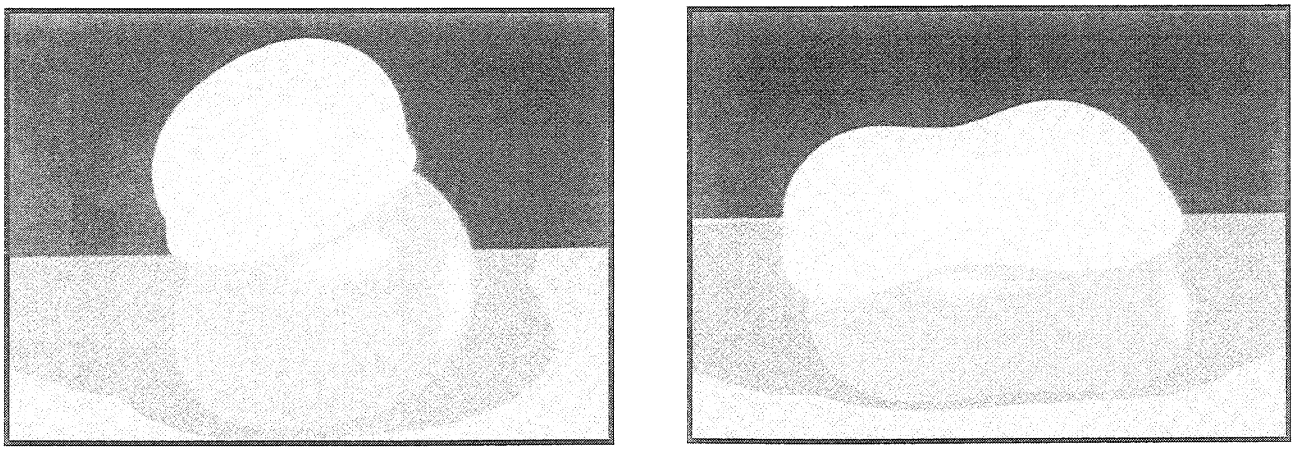


Figure 9 : Collision between two blocks of a stiffer substance.

Conclusion

This paper presents a new method for modeling and animating soft blocks of highly deformable material, capable of breaking into pieces. The model is hybrid: deformable material is composed of several rigid skeletons interacting together, coated with elastic flesh. Our implicit formulation for the flesh offers an analytic definition of the object surface throughout the simulation. Coherent parameter choices for the external and internal structures ensure consistent volume variations during the simulation. Used for precise collision detection, the flesh deforms during contacts with other objects. Reaction forces computed along contact surfaces are transmitted to the internal structure, producing subsequent deformations. Thanks to an efficient adaptive sampling algorithm, the simulation is produced at interactive rates, which facilitates the precise tuning of an animation.

Work in progress includes attempts to further optimize the animation algorithm (in particular by improving the way we recompute and validate seed points), to find a better solution for avoiding volume variations when objects break into pieces or melt, and to implement solid friction between contact surfaces.

Acknowledgements

Many thanks to Nicolas Tsingos for implementing an efficient visualisation of implicit surfaces, to Jean-Dominique Gascuel for his constant support during this research, and to George Drettakis, Kevin Novins, Frederic Cazals and Jean-Christophe Lombardo for rereading this paper.

References

[BW90] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, March 1990.

[Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics*, pages 313–320, August 1993. Proceedings of SIGGRAPH'93 (Anaheim, California, August 1993).

[Hou92] Donald House. Coupled particles: Theory. *Particle System Modeling, Animation, and Physically-Based Techniques (SIGGRAPH'92 course notes Number 16, Chicago, Illinois)*, 1992.

[Jim93] Stéphane Jimenez. Modélisation et simulation physique d'objets volumiques déformables complexes. Thèse de doctorat, Institut National Polytechnique de Grenoble, November 1993.

[LJF+91] A. Luciani, S. Jimenez, J-L. Florens, C. Cadoz, and O. Raoult. Computational physics: a modeler simulator for animated physical objects. In *Eurographics'91*, Vienna, Austria, September 1991.

[LJR+91] Annie Luciani, Stéphane Jimenez, Olivier Raoult, Claude Cadoz, and Jean-Loup Florens. An unified view of multitude behaviour, flexibility, plasticity, and fractures: balls, bubbles and agglomerates. In *IFIP WG 5.10 Working Conference*, Tokyo, Japan, April 1991.

[PB88] J.C. Platt and A.H. Barr. Constraint methods for flexible models. *Computer Graphics*, 22(4):279–288, August 1988.

[TF88] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformations: Viscoelasticity, plasticity, fracture. *Computer Graphics*, 22(4):269–278, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[Gas95] Jean-Dominique Gascuel. Implicit patches: An optimized and powerful ray intersection algorithm for implicit surfaces. In *Implicit Surfaces'95*, Grenoble, France, April 1995.

[LC87] William Lorensen and Harvey Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

[NB93] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6), November 1993.

[NHK+85] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Objects modeling by distribution function and a method of image generation (in japanese). *The Transactions of the Institute of Electronics and Communication Engineers of Japan*, J68-D(4):718–725, 1985.

[OM93] Agata Opalach and Steve Maddock. Implicit surfaces: Appearance, blending and consistency. In *Fourth Eurographics Workshop on Animation and Simulation*, Barcelona, Spain, 1993.

[OM94] Agata Opalach and Steve Maddock. Disney effects with implicit surfaces. In *5th Eurographics Workshop on Animation and Simulation*, Oslo, Norway, September 1994.

[PW89] Alex Pentland and John Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222, July 1989. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989).

[TM91] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3-D models with local and global deformations : deformable super quadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(7):703–714, July 1991.

[WG92] J. Wilhelms and A. Van Gelder. Octree for faster isosurface generation. *Transactions on Graphics*, 11(3):210–227, July 1992.

[WH94] Andrew Witkin and Paul Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics*, pages 269–278, July 1994. Proceedings of SIGGRAPH'94.

[WMG86] B. Wyvill, C. McPheeters, and G.Wyvill. Animating soft objects. *The Visual Computer*, pages 235–242, August 1986.

[WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, pages 227–234, August 1986.

[WW89] Brian Wyvill and Geoff Wyvill. Field functions for implicit surfaces. *The Visual Computer*, 5:75–82, December 1989.

[Wyv93] Brian Wyvill. Metamorphosis of implicit surfaces. *Modeling, Visualizing, and Animating with Implicit Surfaces (SIGGRAPH'93 course notes Number 25, Anaheim, CA, USA)*, August 1993.

[ZJ91] Cornelia Zahlten and Hartmut Jurgens. Continuation methods for approximating isovalued complex surfaces. In *Eurographics'91*, pages 5–19, Vienne, Autriche, September 1991.

Chapitre 6

Combining simulation with trajectory control

Ce travail, qui permet de combiner contrôle de trajectoire et simulation par modèles générateurs, a fait l'objet de deux articles. Le premier a été présenté dans un colloque en 1993. Le second, beaucoup plus complet, vient d'être accepté par une revue internationale à comité de lecture :

[LG93] An Approach for Guiding Colliding Physically-Based Models. Alexis Lamouret and Marie-Paule Gascuel. Dans les actes de *Fourth Eurographics Animation and Simulation Workshop*, Barcelone, Espagne, pages 209-219, septembre 1993.

[LGG95] Combining Physically-based Simulation of Colliding Objects with Trajectory Control. Alexis Lamouret, Marie-Paule Gascuel et Jean-Dominique Gascuel. à paraître en 1995 dans *The Journal of Visualization and Computer Animation.*

Résumé

La méthode que nous présentons est destinée à faciliter l'emploi de modèles générateurs par un animateur, habitué aux systèmes de synthèse du commerce. L'idée de base consiste à lui offrir une interface avec laquelle il est familier, tout en fournissant un module de simulation capable de détecter les collisions, et d'augmenter ainsi le réalisme de l'animation produite.

L'utilisateur définit un ensemble de positions clés pour guider le mouvement, mais n'a pas à se préoccuper d'éviter les interpénétrations, de déformer les objets lors de collisions, ni même du réalisme du mouvement qu'il définit. Le module de simulation permettra de *corriger* cette trajectoire et de calculer les déformations des objets en fonction de leurs propriété physiques (masse, inertie, raideur...), ainsi que des collisions et des contacts détectés automatiquement au cours du mouvement. Pour ce faire, certains des objets sont munis d'effecteurs capables de produire des forces et des couples à partir d'un PD-controlleur généralisé. Cela les rend capables de reprendre leur trajectoire après en avoir été déviés par une action extérieure brutale, comme une collision. La simulation

est interactive, même dans le cas de structures articulées, ce qui facilite la mise au point des paramètres d'une animation.

132

Combining Physically-Based Simulation of Colliding Objects with Trajectory Control

Alexis Lamouret
Marie-Paule Gascuel
Jean-Dominique Gascuel

iMAGIS / IMAG*– INRIA
BP 53, F-38041 Grenoble cedex 09, France
email: Alexis.Lamouret@imag.fr,

December 8, 1994

Abstract

This paper describes a method that facilitates the use of physically based models by animators. The main point is to give the animator a familiar interface, while providing a simulation module which detects collisions thus enhancing realism.

The user gives a set of key-frames to guide motion, but does not have to address problems such as interpenetration avoidance, deformations due to collisions, or realism of motion. The simulator will *correct* the trajectories and compute deformations according to each object's physical properties (such as mass, inertia, stiffness) as well as the collisions and contacts automatically detected during motion. To achieve this, objects are provided with actuators capable of generating forces and torques computed via generalized Proportional-Derivative controllers. When deflected by external actions, actuated objects try to return to their initial path. Speed variations over time are computed during the simulation, and depend on the complexity of the paths, on the objects' models, and on the events such as collisions occurring during motion. In addition simulations are generated at interactive rates, even in the case of complex articulated objects. This facilitates the fine tuning of an animation sequence.

Keywords: animation, simulation, motion control, articulated objects, collisions.

1 Introduction

Over the past few years, physically-based models have demonstrated their usefulness for the automatic generation of realistic motion, including situations where bodies collide or remain in contact. However, these techniques are not yet widely used in industrial animation softwares. The main reason is the kind of interface they offer: the user of a simulation module has to provide a model for each object (with parameters such as mass, inertia, stiffness, etc), initial conditions, and a set of externally applied forces. This

*IMAG is a Research institute affiliated with CNRS, Institut National Polytechnique de Grenoble and Université Joseph Fourier.

interface is clearly inappropriate for an animator, who has a precise idea of the script he wants to follow, but has no way of finding out which forces should be applied to the models over time.

Perhaps we should address the problem in another way : what would an animator want ? It is desirable to take advantage of the help physically-based animation can offer, in particular for automatic collision detection and response, deformation of the objects, and realism of motion. And at the same time, keep the most frequently-used and most convenient interface : key-frames. In addition, throughout the process, the animator should be able to see the results of any parameter change in real time.

This paper presents an approach for combining the physical simulation of objects, including those in collision or contact situations, with trajectory control. The user still defines key-frames to guide motion, but does not have to consider the realism of motion, nor the collisions that will take place. The system *automatically corrects* this rough motion during an interactive simulation process: the final trajectory, the speed variations and the deformations of the objects over time will depend on the physical models and on events such as collisions and contacts detected during motion.

1.1 Related Work

Motion control has become a very important issue in physically-based animation. We briefly review the main approaches.

Constraint methods [BB88, Ove91b, GG94] and *inverse dynamic techniques* [IC87, Dum90, Ove91a, Ove93] offer direct control on some of the objects degrees of freedom. To animate a complex structure composed of different solids connected by hinges, the user can specify trajectories for some of the components, and let the system compute the other component's motions during a physically-based simulation process. Unfortunately, this is not sufficient for attaining our goal: The user has no help for improving the realism of the "leader components" motions, and in particular there is no automatic collision detection and response for these components.

Optimization techniques provide a convenient interface for the user, who defines a set of key-positions for the objects. Physical models are used for finding correct interpolations between these key-positions, during a minimization process (the criteria most frequently used results in the minimization of the amount of energy needed to perform the motion).

In [BN88], the method is restricted to motions expressed by linear differential equations, and to interpolation between an initial and a final configuration. The resolution requires two steps: backwards integration over time of a Ricatti system expressing the final conditions, and then forwards simulation from this result. An attempt for generalizing the method to non-linear systems is described in [Han93], but it requires iterating the process of forwards and backwards integrations over time until it converges. The approach is therefore compute-intensive, and very difficult to handle in complex situations.

In [WK88], discrete objects trajectories are improved during a series of iterations, under a set of space-time constraints defined by the user. Dynamic laws are included as extra constraints between position, velocity and acceleration parameters over time. In spite of the improvement to the method presented in [Coh92] – use of space-time windows to independently recompute parts of the motion without affecting everything – this technique does not produce animation sequences at interactive rates. In addition, the user has to

pre-define interactions between objects as extra constraints that hold during specified time intervals. This can be time consuming (imagine, for instance, the complexity of defining constraints for a deformable wheel rolling and jumping on a bumpy floor). Moreover, the contact characteristics – deformations of the solids, contact duration – are then dictated by the user-defined constraints rather than computed from the physical parameters. This can adversely affect the realism of motion.

Methods based on controllers have the advantage of being compatible with forward simulation techniques, which facilitate the automatic collision detection and response. In many applications in Computer Graphics, a control module which senses some variables as input and outputs a set of actuator forces and torques is combined with a classical physically-based simulator.

Specialized controllers in [RH91, vdPFV92] are dedicated to legged locomotion, and address specific problems such as organizing leg actions and maintaining balance. [JLR93] uses controllers to maintain speed constraints on land vehicles interacting with various landscapes. [DLC93] uses "primary muscles" to generate motion of objects either controlled in real time by an operator, or moving autonomously, for a purpose of path planning.

Adequately tuning a controller for each model in the scene represents a great deal of work, and requires specialized knowledge. Recently, some approaches have been developed for automatically generating a controller for a given model, according to a criteria such as "cover the longest distance in a given time interval". In [vdPF93], a suitable controller is selected during a random generation process. A genetic algorithm is used to perform the same goal in [NM93]. These approaches are promising for enabling a system to automatically generate different kinds of locomotion, whatever the object may be. But they are not aimed at producing a specific motion that is close to a script or to key-frames defined by an animator.

1.2 Overview

This paper describes a general method for controlling physically-based models according to a predefined script, while preserving their ability to automatically detect and respond to collisions. The animator specifies key-frames to guide motion, but does not have to address problems such as realism of motion or interpenetration avoidance. The system corrects this rough motion during a forward simulation over time, performed at interactive rates.

To achieve this, objects are provided with actuators of limited strength, representing motors or muscles. At each animation step, actuator forces and torques are computed from generalized Proportional-Derivative (PD) controllers that use the difference between a target position and the current object position as their main input. Targets are located on interpolation curves defined from the animator's key-positions (see Figure 1). Their motion is closely related to the associated object motion. When an object is deflected by a collision, its target position may remain unchanged during a few time steps in order to avoid a loss of precision in the trajectory control.

Section 2 explains the basic principles of the approach by addressing the case of an isolated solid guided by key positions and orientations defined by an animator. Section 3 deals with more complex situations. The controlled objects can be connected to others

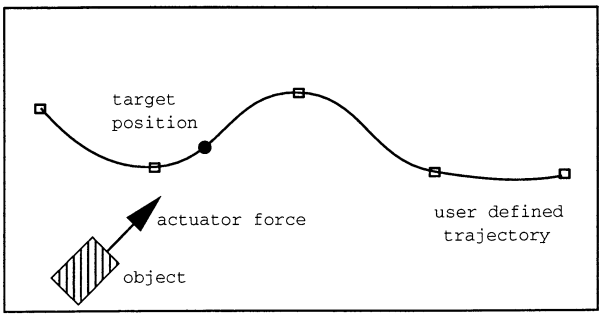


Figure 1: Actuated object guided along a path

by various kinds of hinges. Each object may be subject to a set of external actions, which may or may-not last over time. Section 4 describes results, and shows that the use of our method greatly simplifies the animator's task. Section 5 presents our conclusions and discusses works in progress.

2 Basic Trajectory Control

In the remainder of this paper we use the following notation, where rotation speed matrix and angular speed vector are linked together by: $\forall \vec{a} \in I\!R^3 \quad \tilde{\omega}.\vec{a} = \vec{\omega} \wedge \vec{a}$

Symbol	Description
m	mass
J	inertia tensor
$\vec{F}$	force
$\vec{T}$	torque w.r.t. the center of mass
$\vec{v}$	linear speed vector
$\vec{\omega}$	angular speed vector
$\tilde{\omega}$	rotation speed matrix
$\vec{x}$	current translation vector
R	current orientation matrix

This section studies the case of a single isolated object, which is guided by animator-specified key-frames. More complex cases such as objects subject to lasting external forces and components of articulated structures will be treated in Section 3. According to the desired type of control, the object is provided with a translation actuator, a rotation actuator, or both. Each actuator is characterized by a limited strength: the force or torque it can generate is limited to a specified threshold.

The simulation method looks like a game of cat and mouse. At each time step, the object tries to move towards a target position located on the user-defined path. But this target moves, and its speed is closely related to the object motion. The next two sections detail the key points of the method: how to compute the actuator action from the current target position, and how to modify this position at each time step.

2.1 Computing Actuator Forces and Torques

Two kinds of actuators are available in our system. Translation actuators limited in strength generate a translation force $\vec{F}$ applied to the object's center of mass, verifying: $||\vec{F}|| < F_{\max}$. Rotation actuators generate a torque $\vec{\mathcal{T}}$ verifying: $||\vec{\mathcal{T}}|| < \mathcal{T}_{\max}$. The principle of the actuator action is to generate forces or torques to move the object towards an associated target position $\overrightarrow{x_{\text{target}}}$ or orientation R_{target}.

Suppose that there is no external action. According to the the Newton integration scheme described in Appendix A, the force to apply in order to reach the position $\vec{x}_{\text{target}}$ in one time step dt is:

$$\vec{F}(t) = 2m\,\frac{\overrightarrow{x_{\text{target}}} - \vec{x}(t)}{dt^2} - 2m\,\frac{\vec{v}(t)}{dt} \tag{1}$$

Unfortunately, this force is not suitable in our case, for many reasons. First of all, the velocity needed to reach the target in one time step would propel the object far past the target in subsequent time steps. Next, entirely suppressing the effect of the previous speed-vector would effectively suppress the inertia, as long as the maximum strength of the actuator is not reached. Finally, most simulation systems use an adaptive time step to regulate the simulation when problems such as overly deep interpenetrations between colliding objects occur. Using the current time step for computing the actuator action would cause unexpected acceleration or deceleration for each variation of dt.

Our approach is different. We compute a force so that the object will comply a given portion α of the distance to the target in a fixed relaxation time Δt (to avoid any problems with the adaptive time step). A parameter β is used for taking the speed of the object into account, without entirely compensating for it. This leads to the formula:

$$\vec{F}(t) = 2m\,\frac{\alpha\left(\overrightarrow{x_{\text{target}}} - \vec{x}(t)\right)}{\Delta t^2} - 2m\frac{\beta\vec{v}(t)}{\Delta t} \tag{2}$$

- If $\beta = 0$, the speed of the object is ignored during the computations. The object will then oscillate indefinitely around the target position, since there is no absorption. In practice, this limit motion is obtained only for a very small simulation time-step. Due to integration errors, the trajectory quickly diverges with usual time steps.
- $\beta = 1$ means we entirely correct effect of speed, hence suppressing inertia.
- In between we can tune β to obtain a more or less damped trajectory, as shown in Figure 2. The value of β therefore affects the rate of convergence to the target position. An analogy with the control theory, which will be further developped in Section 2.3, gives a theoretic value for β's critical damping value:

  $$\beta_{\text{critical}} = \sqrt{2\alpha}$$

  During our experiments, we have verified that this heuristic-value for β_{critical} is good for sufficiently small time steps.

In practice, an animator may not want to directly give a value for β, but rather to tune the system between more or less damped motions. The animator thus gives an "elasticity

coefficient" $B \in [0, 1/\beta_{\text{critical}}]$, with default value at 1, and we use $\beta = B.\beta_{\text{critical}}$. When a high convergence speed is desired, the choice of a B lightly smaller than 1, and depending on the convergence criterion is recommanded. For instance, experiments show that if we want to converge quickly while staying within one tenth of the initial distance from the trajectory, a good choice is $B = 0.7$.

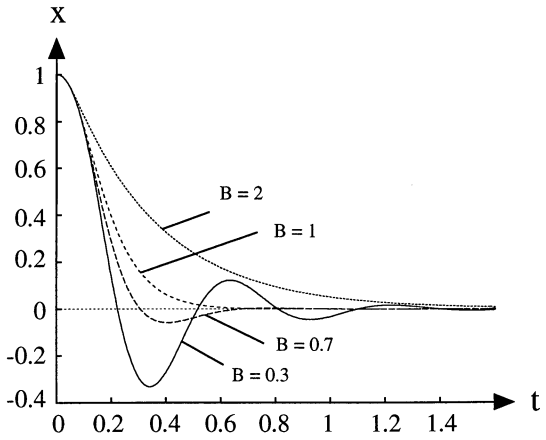


Figure 2: Overdamped, underdamped and critical curves as a function of B, for a dynamic point trying to reach a stationary target position, with $\alpha = 0.1$.

The computations for a rotation actuator are similar (see Appendix A for the Newton integration scheme for rotations). We compute the torque which would rotate the object by a percentage α towards a target orientation during the relaxation time Δt, with a compensation of rotation speed regulated by a parameter β:

$$\vec{T}(t) = \frac{2.J\left(\alpha\vec{W} - \beta\vec{\omega}(t)\right)}{\Delta t} - \vec{\omega}(t) \wedge J\vec{\omega}(t), \tag{3}$$

$$\text{where} \quad \tilde{W} = \frac{R_{\text{target}} R_{\text{object}}^{-1} - I}{\Delta t}. \tag{4}$$

Let us now explain how target positions and target orientations are computed from the user-defined key-frames and from the associated object motion.

2.2 Computing a New Target Position

As emphasized in the introduction, the animator roughly specifies the trajectory of actuated objects through a set of key positions and/or orientations. This data is first interpolated to get a continuous trajectory for the target. Methods for interpolating between positions and orientations (converted into quaternions) can be found in [Duf86]. Target values for computing both forces or torques applied by actuators at each time step are represented by current scalar parameters along spline curves.

Moving the target position at a constant speed along the curve would not be a good solution. Indeed, the actuated object can be slowed down by a collision, and will not follow the predefined path with sufficient precision if the target has run too far ahead.

Moreover, the actuator callback force (or torque) would become too large in such a case, and therefore produce unwanted speed variations for the object.

We instead compute the next target position from a *base distance* specified by the user. The system ensures that any point of the predefined path will be at most at the base distance to the object final trajectory.

The idea consists in trying to always keep the target at the base distance from the object, with the constraint that the target never moves backwards along its path. Then, if an object is pushed backwards, its target position remains the same during a few time steps. If the object goes fast, so will the target.

To achieve this, we define a distance function to be used between the object and the target. For translation, it is simply the metric distance, wereas for orientations we use quaternions [Sho85] as follows: The distance between two quaternions q_a and q_b is defined by:

$$d(q_a, q_b) = angle(q_a.q_b^{-1}) = 2acos(Re(q_a.q_b^{-1}))$$

At each time step, we consider the last position of the target and then look for the first higher parameter value for which the distance is reached (this is done by binary search).

2.3 Comparison with Proportional Derivative (PD) Controllers

The computations used in Section 2.1 for the actuator forces and torques are quite similar to the action of a Proportional Derivative controller [Sev89].

For instance in the case of forces, a PD controller sensing an error $e = \overrightarrow{x_{\text{target}}} - \vec{x}(t)$ and its derivative $\dot{e} = \overrightarrow{v_{\text{target}}} - \vec{v}(t)$ would produce a force $\vec{F} = \alpha e + \beta \dot{e}$ aimed at minimizing e. This force, which differs from the force of equation (2) by addition of the term $\beta \overrightarrow{v_{\text{target}}}$, would tend to make the object reach the target and regulate its speed based on the target speed.

This formula would not work for our application. Our aim is to make the target regulate its speed on the object motion, rather than the contrary. We do this by always leaving the target at the same distance from the object, with the result that the object-target system will reach a constant speed on a straight line when no other force is observed. Conversly, if we used the force formula given by the PD controller, the object would accelerate at each time step to try to reach the target, so the target would also accelerate at each time step, to keep its distance.

Consequently, our method can be seen as a generalized version of a PD controller, adequately adapted to our specific goal.

3 Generalization to Complex Situations

The method just described enables the combination of physically-based simulation of an isolated solid with trajectory control. The actuator force described above is still sufficient if an external force is applied to the object during a few time steps. For instance, when a collision with another body is detected, the controlled object is deflected from its trajectory, but comes back closer to the predefined path after a while, due to the action of its actuators.

However, most of the objects used in Computer Animation are subject to continuous external actions such as gravity. Some external actions may be very complex, such as

the interactions between neighboring components in an articulated structure. Hence, the set of externally applied forces and torques must be adequately taken into account while controlling the motion.

This Section first describes a method for animating articulated structures that is well suited to our approach for control. Then, the control algorithm presented earlier is adapted to general situations, where various kinds of external actions may be applied on each solid.

3.1 Animation of Complex Structures

We are looking for a convenient method for animating articulated bodies, and more generally constrained structures. This method must be well suited to the approach for trajectory control we have developed. In particular, the user must be free to independently associate actuators to only some of the component of a structure, the other ones being animated by another algorithm. This forbids the use of approaches based on the expression of laws of dynamic in the parameter space of articulated objects. Another important criteria is the efficiency of the whole animation system. As we said previously, the simulations should be computed at interactive rates. To achieve this, we use the "*displacement constraints*" method detailed in [GG94]. We present here only a brief review of the technique.

Complex articulated structures are built from independent solid components connected together by geometric constraints. The user is free to choose the number of degrees of freedom in rotation and in translation at hinges, and to specify angular or linear constraints on motion. The graph of constrained objects may contain any number of closed loops. A technique for the automatic construction of valid initial positions is provided together with the animation method.

At each computation step, the solids move first as if they were independent according to Newton integration scheme detailed in Appendix A. Then, constraints are met through *iterative tunings in displacements.*

Displacements associated with a single constraint

Let us first present the method in the case of a single constraint between two solids. We will explain how to combine the action of multiple constraints later on.

A "point to point" constraint between two solids S_1 and S_2 creates a joint with 3 degrees of freedom in rotation by making two points, P_1 of S_1 and P_2 of S_2, coincide throughout movement. See Figure 3.

Meeting a "point to point" constraint by translating a solid without any rotation is always possible, but would most of the time produce unrealistic behaviors. To find the adequate proportion between rotation and translation we make an analogy with the action of some hypothetical rubber-band that would have maintained the constraint during a time step.

This leads to the following computation scheme, where the proportion between rotation and translation, and the relative displacements of the two solids are consistent with their parameters of mass and inertia :

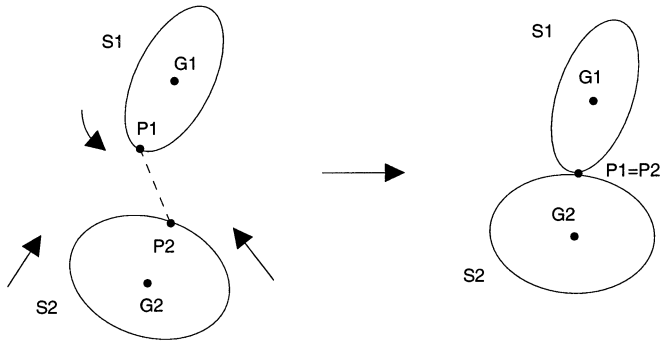


Figure 3: A "point to point" constraint between two solids.

1. Apply respectively to S_1 and S_2 the small rotations given by the rotation vectors:

$$\begin{cases} \overrightarrow{\Delta R_1} = \frac{m_1 m_2}{m_1+m_2} J_1^{-1} \left(\overrightarrow{G_1 P_1} \wedge \overrightarrow{P_1 P_2}\right) \\ \overrightarrow{\Delta R_2} = \frac{m_1 m_2}{m_1+m_2} J_2^{-1} \left(\overrightarrow{G_2 P_2} \wedge \overrightarrow{P_2 P_1}\right) \end{cases} \tag{5}$$

   where m_i are the masses and J_i the inertia tensors of the two solids.

2. Then, from the positions P_1', P_2' of the points *after rotation*, compute and apply the small translations exactly satisfying the constraint:

$$\begin{cases} \overrightarrow{\Delta x_1} = \frac{m_2}{m_1+m_2} \overrightarrow{P_1' P_2'} \\ \overrightarrow{\Delta x_2} = \frac{m_1}{m_1+m_2} \overrightarrow{P_2' P_1'} \end{cases} \tag{6}$$

The method can be extended in order to offer some translational degrees of freedom at hinges, possibly limited in scope. To constrain the movement of P_1 in any sub-area defined w.r.t S_2, P_2 is replaced in equations (5) and (6) by the point P_2' of the chosen domain which is the closest from P_1. "Point to segment", "point to curve", "point to surface", "point in sphere" are examples of simple constraint concepts that may be useful.

In the method presented so far, each constraint leaves 3 degrees of freedom in rotation between the two solids. Restricting rotations at hinges may be useful, and can be done by the same type of approach. Once a parametrization (such as quaternions) has been chosen for orientations, one can compute the smallest rotation to bring the angular "distance" between two objects back to some allowed freedom space. This rotation is split between the solids according to their respective inertia for this particular axis.

Combining displacements due to individual constraints

In practice, several constraints can simultaneously be applied to a solid, so rotations and translations due to each constraint must be combined together. A sum of the displacements computed for each individual constraint would conserve first order momenta, but would lead to divergences in some particular cases, as shown in [GG94].

This problem can be solved by weighting the displacements affected to the solids before summing them. In order to obtain the conservation of first order momenta, the same weighting factor must be used for couples of displacements due to the same constraint. If S_i

and S_j are two objects linked by a given constraint, we weight the associated displacements by: $\frac{1}{\max(n_i,n_j)}$, where n_i and n_j are the numbers of constraints respectively applied on each solids.

A series of iterations is needed to fulfill the constraints as soon as more than one constraint per solid are applied. This is done through an iterative process which stops when all the resulting corrections are smaller than a specified threshold, or when a maximal number of iterations is reached. Limiting the number of iterations avoids deadlocks when the system is over constrained. The equations used lead to very fast convergence rates. Usually a few iterations are sufficient, even when the graph of constrained objects contains closed loops (see [GG94]).

Correcting the kinematic behavior of constrained solids

The corrections due to constraints must be taken into account in the kinematics of movement, as if we had added constraint forces. Once a position of the solids that meets the constraints have been found, we adjust their linear and angular speeds by considering the positions and orientations they have effectively reached during a time step:

$$\begin{aligned} \vec{v}(t+dt) &= 2(\vec{x}(t+dt) - \vec{x}(t))/dt - \vec{v}(t) \\ \tilde{\omega}(t+dt) &= 2\left(R(t+dt)R^{-1}(t) - I\right)/dt - \tilde{\omega}(t) \end{aligned}$$

In conclusion, this animation algorithm for complex structures is simple and efficient. It basically gives the same effect than the computation of a set of constraint forces at hinges[1]. However, tuning displacements is more direct, as there is no need of integrating constraint forces during the series of iterations needed for fulfilling constraints. Quite general constrained structures can be built and animated with this method, with no restriction on the particular algorithm used for computing the independent motions of each object. Therefore, this approach seems very well suited to our trajectory control module.

3.2 Control of Objects Submitted to Any External Action

As emphasized earlier, the study of isolated objects is very restrictive. Most objects are subject to various kinds of external actions during an animation sequence. Some of these actions result in continuous external forces such as gravity or fluid friction, while others only last for a few time steps such as most collision forces. Objects which are components of articulated structures are subject to highly varying small interactions maintaining constraints with their neighbors.

To keep the associated object close to the predefined-path, the control module must take the effect of external actions into account in the computation of actuator forces and torques. Typically, an object subject to a constant gravity force must compensate for it with its actuator action, otherwise it will remain far below the goal trajectory, as shown in Figure 4.

[1] although it is not proved that the methods are equivalent

142

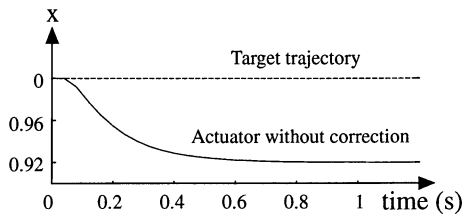


Figure 4: Effect of gravity for a ball following a target on a straight line ($\alpha = 0.1, \beta = 0.6$)

However, when an animator introduces animation effects such as an object bouncing on obstacles due to gravity and to response forces, he does not want the controle module to instantaneously hide these effects. An object should not react too quickly to an externally applied force, but still be deflected by sudden collisions. After a reasonable period of adaptation, the object should tend to compensate for an externally applied force which remains constant over time. In particular, an object subject to gravity should be controllable.

First of all, we need to provide the control module with an adequate sensor, able to apprehend the effect of external actions. Directly sensing the value of external forces would not be a good solution. Due to our approach for animating complex structures, some of the external actions may be directly expressed by displacements rather than by forces. So we use the object current position, orientation and speed, as entries for the controller.

At each time step, the controller needs to approximate the sum of external actions during the last time interval. This can be done by comparing the object current location with the "predicted position" it was trying to reach at the last time step. In practice, we prefer to use speeds rather than positions and orientations for approximating the sum of external actions. This reduces numerical errors, as we avoid the approximation resulting from the conversion of a rotation vector into a matrix. The evaluated external forces and torques are:

$$\vec{F}_{\text{ext}}(t - dt) = m\frac{\vec{v}(t) - \vec{v}(t - dt)}{dt} - \vec{F}_{\text{actuator}}(t - dt) \tag{7}$$

and similarly for torques:

$$\vec{\mathcal{T}}_{\text{ext}}(t - dt) = J\left(\frac{\vec{\omega}(t) - \vec{\omega}(t - dt)}{dt}\right) + \vec{\omega}(t) \wedge J\vec{\omega}(t) - \vec{\mathcal{T}}_{\text{actuator}}(t - dt) \tag{8}$$

Overcoming the action of these external forces and torques could be done by adding the opposite of $\vec{F}_{\text{ext}}$ or $\vec{\mathcal{T}}_{\text{ext}}$ to the actuator action. But this would not be a good idea for several reasons. As said before, we do not want to correct too quickly the action of external forces. Another point is that articulated objects are often subject to highly varying small displacements for maintaining constraints, due to very complex interaction forces between neighboring components. Using – with a delay – the opposite of these forces can produce an amplification phenomenon which leads to oscillations around the goal trajectory. Such a situation is depicted in Figure 5 (a), which represent the horizontal oscillations generated during the controlled motion of a three-link articulated arm whose extremity should move down on a vertical line.

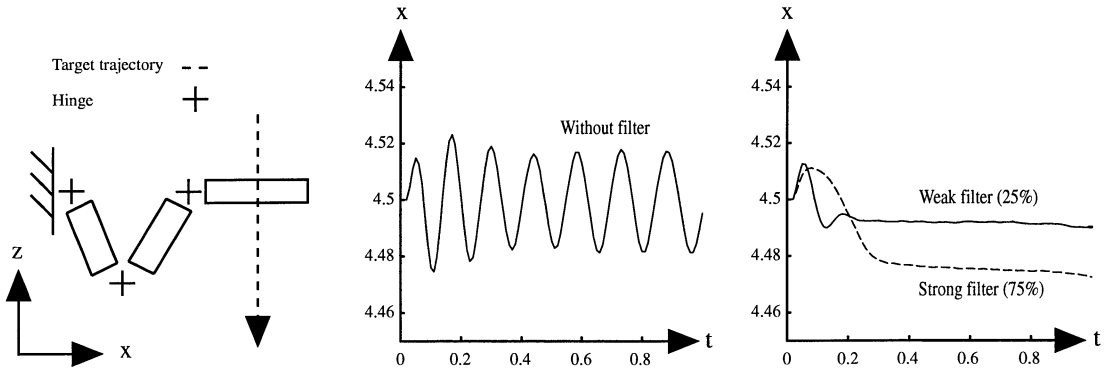


Figure 5: Filter effect on an articulated object.
(a) motion without filtering the actuator forces.
(b) motion with a filter.

The solution consists in using a filter for computing actuator forces and torques from approximated external actions. The first benefits of this filter is the production of softly varying actuator forces (Figure 5 (b) shows that the oscillations disappear). It also provides a convenient parameter for regulating the delay between external action and response of the control module.

In order to offer an amount of filtering independent of the adaptive time steps used during the simulation process, the user specifies a parameter γ representing the amount of filtering per second. The actuator correction force or torque is then computed from the approximated external action and the previous values of corrections at $t - dt$ by using the filter parameter γdt:

$$\vec{F}_{\text{corr}}(t) = -\gamma dt \vec{F}_{\text{ext}}(t - dt) + (1 - \gamma dt)\vec{F}_{\text{corr}}(t - dt) \quad (9)$$

$$\vec{\mathcal{T}}_{\text{corr}}(t) = -\gamma dt \vec{\mathcal{T}}_{\text{ext}}(t - dt) + (1 - \gamma dt)\vec{\mathcal{T}}_{\text{corr}}(t - dt) \quad (10)$$

Finally, the complete algorithm for computing actuator forces and torques is:

1. Compute the main component of the actuator action from equations (2) and (4).
2. Compensate from observed external actions by adding the correction terms given by equations (9) and (10), where $\vec{F}_{\text{ext}}$ and $\vec{\mathcal{T}}_{\text{ext}}$ are computed from (7) and (8).
3. If necessary, truncate the actuator forces and torques according to the maximal available strength.

4 Results

4.1 Animation Background

We have implemented the trajectory control method within the framework of the animation system described in [Gas93, GG94]. Each solid, either rigid or elastic, is structured in:

- A rigid component, which can be stationary, follow a predefined path, or be dynamically animated from rigid body equations of motion of Appendix A. When the

object is composed of elastic material, the mass and inertia tensor of the rigid layer are computed from the object rest shape.

- A coating layer at rest with respect to the rigid component, which represents the material constituting the object. Coating layers give the geometry of the object surface. Our model for elastic coatings, based on an implicit formulation, generates exact contact surfaces between colliding objects and provides precise evaluation of response forces.

The animation algorithm is composed of the control module previously described, and a simulation module consisting of three steps: Integration of the individual equations of motion for each solid rigid component, treatment of constraints through iterative tunings in displacement, and collision detection and response computed from the coating layers. The whole simulation process is based on an adaptive time step: if the time step used is too large with respect to the objects motion, divergences could appear in the constraint processing and/or in the collision module. When this problem is detected, the system goes back in time with a smaller time step as depicted in Figure 6. When no more exception message are generated during a fixed number of simulation steps, the system increases dt, if it does not exceed the display rate.

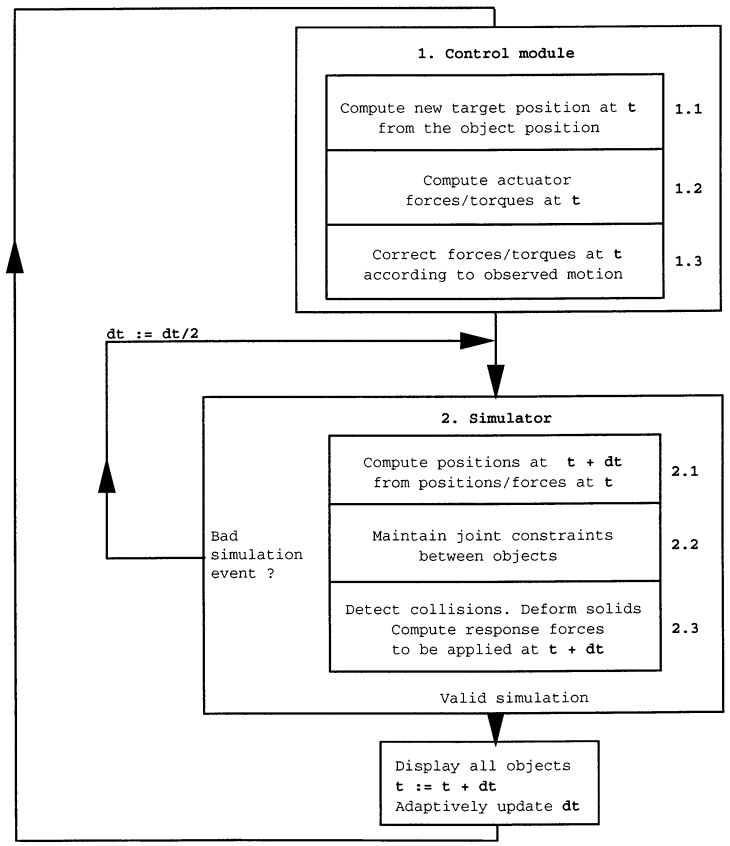
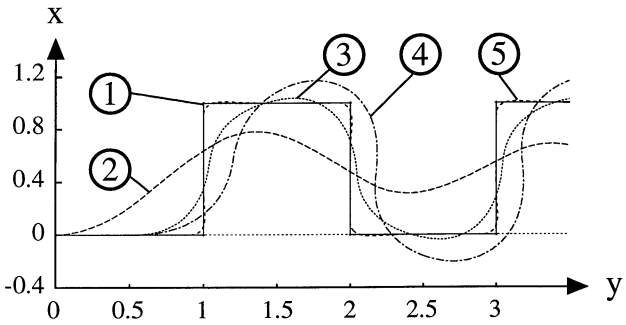


Figure 6: The animation algorithm

4.2 Examples in 2-D

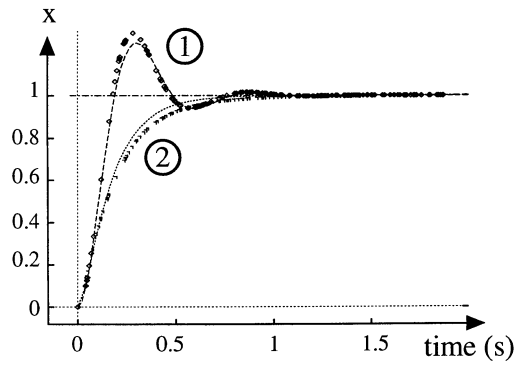
Before showing animation sequences with a complex environment, let us present some results in 2-D to show how the tuning of the different parameters enables to obtain various motions.

Smoothing a rough trajectory



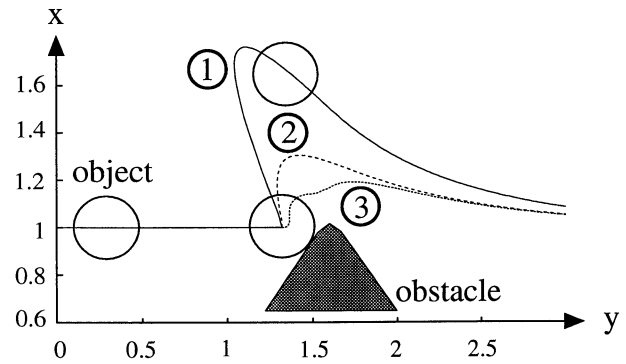
This figure shows various motions for a trajectory physically impossible (curve 1). Changing the base distance (curves 2 and 5) enables to choose how much we want to smooth the trajectory. Changing α within a certain range, and with the same B affects only the speed of the motion. Changing B (curves 2, 3 and 4) affects the damping of motion.

Validation of the method with random time step



The method will be used in an animation system with an adaptative time step. It is therefore important to have a good tolerance to random variations of the time step. This figure shows the motion of the object for a stationary target position, for a constant time step (curves) and for a random time step (dots). We see that the result is quite the same in both cases ($\alpha = 0.1$: $B = 0.5$ in (1), $B = 1.2$ in (2)).

Samples of collisions



In this figure an actuated object collides with a rigid obstacle. The trajectory given by the animator is a simple horizontal line passing through the obstacle. We see various behaviours depending on the mass (1 and 2) and the stiffness of the object (1 and 3). In (3) the object deforms and slides around the obstacle, while in (1) and (2) it bounces more or less violently.

4.3 Simply Implicit

The animation "Simply Implicit" shows the results of our trajectory control method in the case of *a simple object controlled in translation*: an elastic ball subject to gravity and to interactions (that include elastic response and friction) with other solids. The animation was designed for the presentation of the implicit elastic material developed in [Gas93].

The script drawn by an animator (Figure 7) shows the ball bouncing into a nursery, and colliding with a set of flexible toys until it succeeds in knocking them off a shelf.

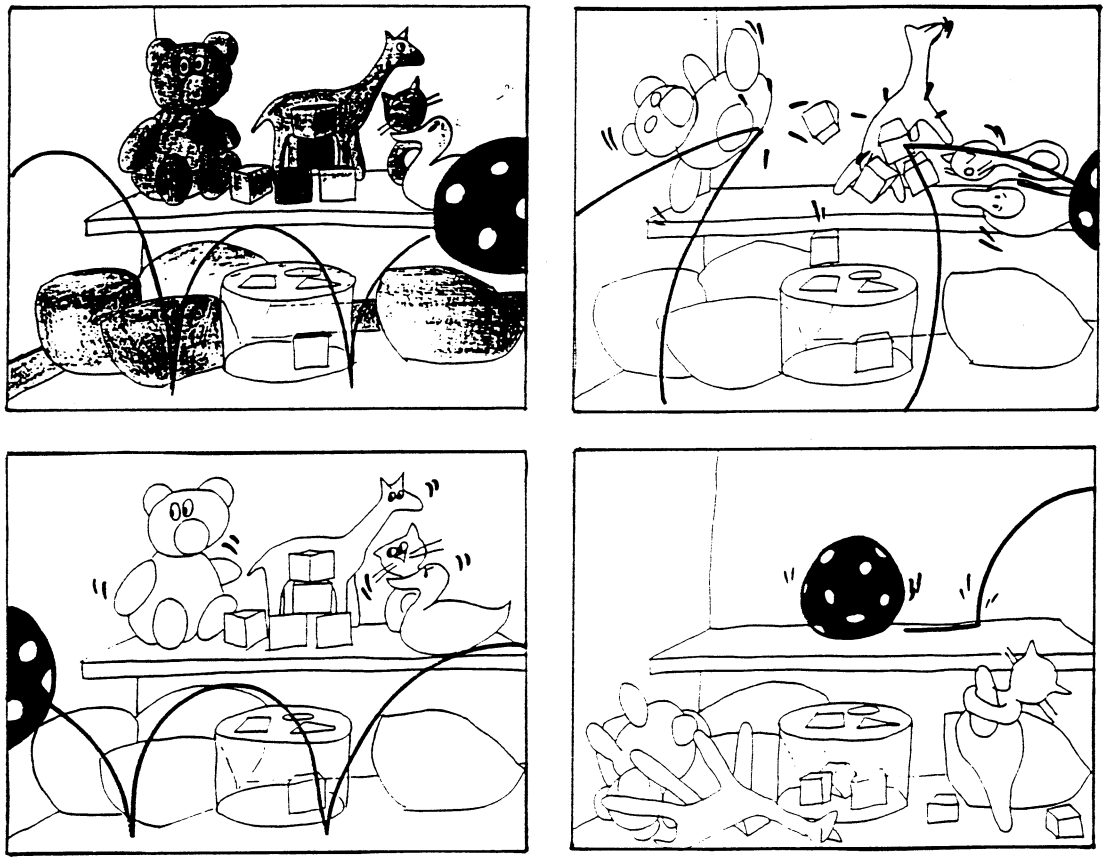
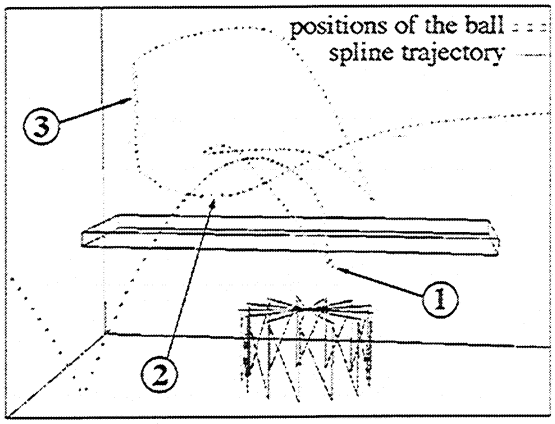
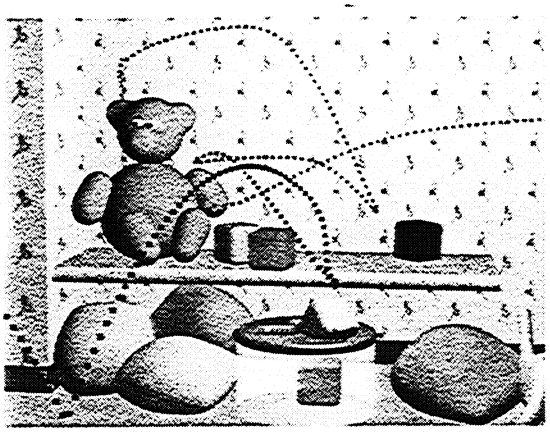


Figure 7: Script drawn by an animator

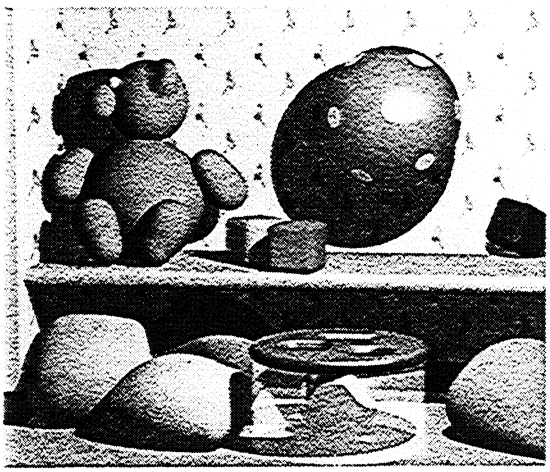
To obtain the final animation, the ball model is provided with a translation actuator. No trajectory control is applied to the other objects, composed of elastic implicit material, and animated by direct use of the simulation module. The animator defines a rough trajectory for the ball by giving some key positions. During this process, he does not have to consider at all the physical properties of the ball (mass, inertia tensor, stiffness coefficient), nor the numerous collisions and contacts that will take place during motion. The system automatically *corrects* the user trajectory during the simulation, according to all these parameters (see Figure 8). In particular, the rotation of the ball is entirely due to the action of friction forces during collision processing.



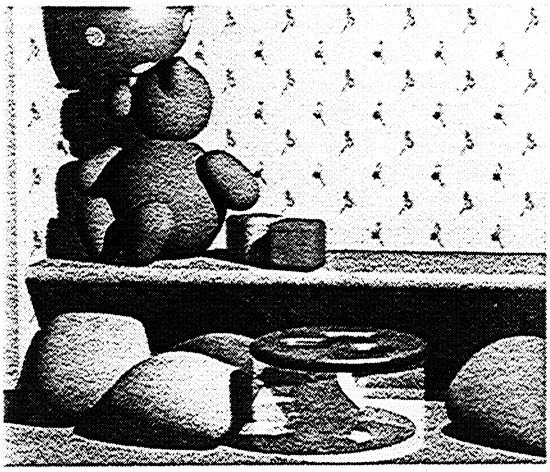
a : Trajectories of the ball and its target.
(1), (2) and (3) show corrections on the trajectory due to collisions.



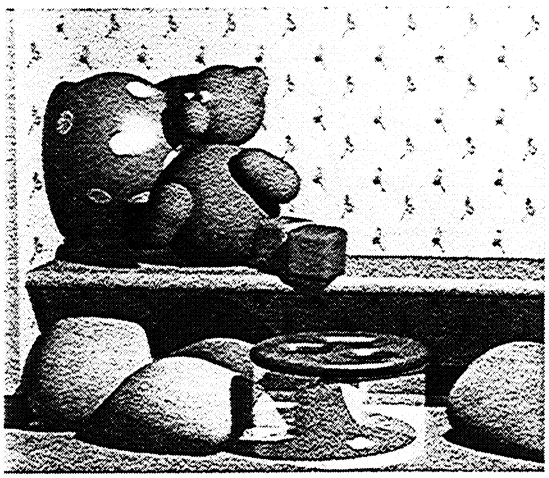
b : Trajectory of the ball in the complete environment.



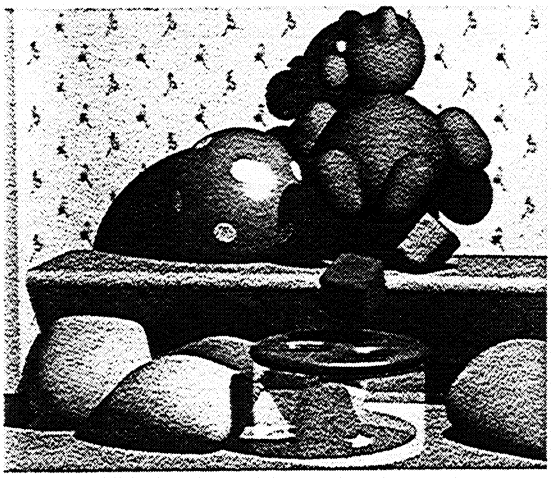
c : The ball enters the scene...



d : ...collides with the wall and the bear...



e : ...pushes the bear off the wall...



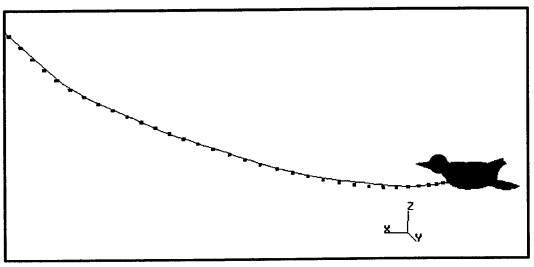
f : ...and starts rolling on the shelf.

Figure 8: Simply Implicit

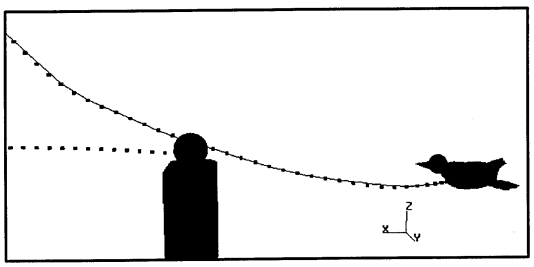
4.4 Bird Flight

Figure 9 shows experiments for the motion of *an articulated object controlled with four actuators*. The object (a bird) is composed of a rigid body and of two rigid wings, connected to the body by hinges with angle constraints. The body is provided with translation and rotation actuators, and the wings with rotation actuators only. Thanks to the "displacement-constraints" module maintaining joints constraints between the body and the wings, no translation actuator is needed for the wings. Up and down key-rotations are given for the wing orientations in order to simulate the bird flight.

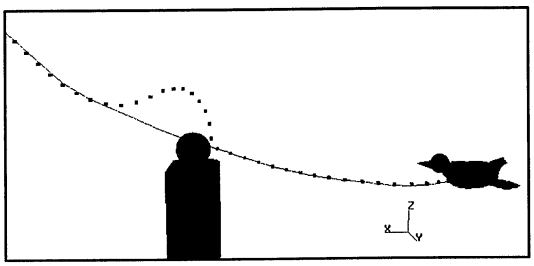
The different experiments show how the bird trajectory is deflected by various collisions (The figure shows only the trajectory of the body in translation).



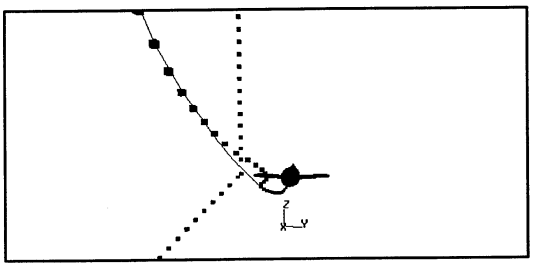
3.1. The bird alone : it is just slightly deviated because of inertia.



3.2. The bird and a light ball : only the ball is asffected by the impact.



3.3. The bird and a very heavy ball : the bird is deviated while the ball remains still.



3.4 The bird and a falling ball : both are deviated.

■■■■■ positions of the bird and the ball at each time step

——— interpolation spline between key positions

Figure 9: Bird flight variations according to different external actions

4.5 Snake

Figure 10 shows a snake composed of nine deformable links connected by hinge constraints. A translation trajectory has been specified for an actuator in translation in the head of the snake, passing through a deformable cylinder. We see the body of the snake unfolding from its initial position and then colliding the cylinder, thus slowing down while dragging it..

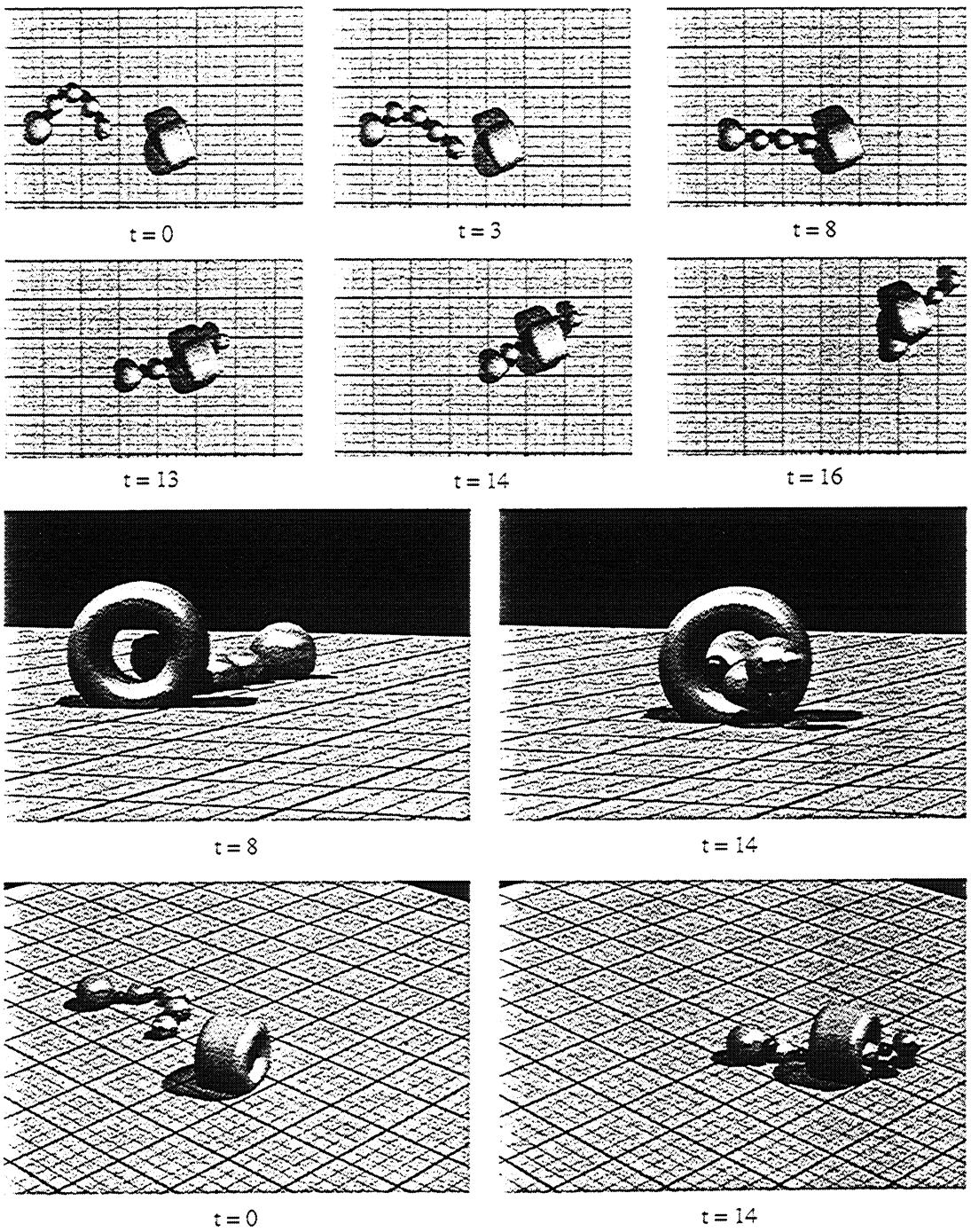


Figure 10: A snake interacting with floor and a deformable cylinder

5 Conclusion

The method developed here should greatly simplify the use of physically-based animation, offering important help for the design of realistic motions and deformations when precise scripts are specified. The aim here is not to compute some muscle action to perform a purely dynamic motion, but rather to increase the realism of a trajectory that may be totally unfeasible without external help. Our method enables objects to follow a pre-defined script which can be as far as true dynamics as the animator wants it to, while adding dynamic quality wherever possible.

A convenient interface is offered to the animator, who controls motion by defining key frames. During this process, there is no need to spend time for carefully tuning the trajectories in order to improve realism, nor for problems such as interpenetration avoidance, or deformations of colliding objects. During an interactive simulation over time, the system will *automatically correct* the trajectories according to the physical parameters and to the collisions and contacts detected during motion. Adequate deformations of colliding objects will be generated during the same process.

The method works by associating translation or rotation actuators (or both) to the objects to control. The control module used for computing the actuator action from the user-defined key-frames derives from a generalized version of PD controllers. Objects can take observed external actions into account while regulating their motion. Deflected by sudden collisions, objects tend to compensate for continuous external actions after a period for adaptation specified by the user. The control method still works in complex situations where objects are components of articulated structures. In addition, trajectory control can be applied to only some of the objects in the scene, pure dynamic simulation or any other algorithm being used for the others. This should help the animator to only focus on the important motions. An object may be controlled in translation but not in orientation (or the opposite), leaving the simulator to generate realistic rotations according to friction forces during collisions and contacts with other objects.

Work in progress

In the method presented here, the velocity of the objects is adjusted during the simulation according to parameters such as mass and inertia, strength of actuators, path complexity, and events such as collisions which can accelerate the motion or slow it down. This greatly improves the realism of motion. However, we are currently studying an extension enabling the user to specify, if needed, preferencial speeds at some of the key-positions. The system will enforce an object to slow down or accelerate during motion, according to the animator specification.

In the current version of the system, target trajectories are independently defined for each solid component, so there is no synchronization between the different actuators acting on an articulated body, nor between different body's motions. Work in progress includes attempts to use a finite-state graph layer to add synchronization constraints to the system. The method will be based on the first extension, since acceleration or deceleration commands are needed to synchronize different motions.

Acknowledgements

We would like to thank Morgane Furio, from the Ecole Nationale Supérieure des Arts Décoratifs (Paris, France) for the design and scenario of the animation "Simply Implicit". Many thanks to Kevin Novins and to George Drettakis for re-reading the paper.

References

[BB88] R. Barzel and A. Barr. A modeling system based on dynamic constraints. *Computer Graphics*, 22(4):179–188, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[BN88] L. Shapiro Brotman and A. N. Netravali. Motion interpolation by optimal control. *Computer Graphics*, 22(4):309–315, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[Coh92] M. Cohen. Interactive spacetime control for animation. *Computer Graphics*, 26(2):293–302, July 1992. Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 1992).

[DLC93] Y. Delnondedieu, A. Luciani, and C. Cadoz. Physical elementary component for modeling the sensori-motricity: the primary muscle. *Fourth Eurographics Animation and Simulation Workshop*, September 1993.

[Duf86] T. Duff. Splines in animation and modeling. *State of the Art in Image Synthesis (SIGGRAPH'86 course notes Number 15, Dallas, TX)*, 1986.

[Dum90] G. Dumont. Animation de scènes tridimensionnelles : la mécanique des solides comme modèle de synthèse du mouvement. *Thèse de Doctorat*, Université de Rennes I, May 1990.

[Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics*, pages 313–320, August 1993. Proceedings of SIGGRAPH'93 (Anaheim, California, August 1993).

[GG94] M.P. Gascuel and J.D. Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer*, March 1994. A first version of this paper appeared in the *Third Eurographics Workshop on Animation and Simulation*, Cambridge, UK, Sept 92.

[Han93] Gabriel Hanotaux. Techniques de contrôle du mouvement pour l'animation. Thèse de doctorat, École Nationale Supérieure des Mines de Saint-Étienne, Université de Saint-Étienne, April 1993.

[IC87] P.M. Isaacs and M.F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics*, 21(4):215–224, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

[JLR93] S. Jimenez, A. Luciani, and O. Raoult. Physical simulation of land vehicles with obstacle avoidance and various terrain interactions. *The Journal of Visualisation and Computer Animation*, 4:79–94, 1993.

[NM93] J.T Ngo and J. Marks. Spacetime constraints revisited. *Computer Graphics*, August 1993. Proceedings of SIGGRAPH'93.

[Ove91a] C. Van Overveld. The generalized display processor as an approach to real-time interactive 3-D computer animation. *The Journal of Visualization and Computer Animation*, 2:16–25, 1991.

[Ove91b] C. Van Overveld. An iterative approach to dynamic simulation of 3-D rigid-body motions for real-time interactive computer animation. *The Visual Computer*, 7:29–38, 1991.

[Ove93] C. Van Overveld. Building blocks for goal-directed motion. *The Journal of Visualization and Computer Animation*, 4:233–250, 1993.

[RH91] M. Raibert and J. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics*, 25(4):349–358, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

[Sev89] Y. Sevely. *Systèmes et asservissements linéaires échantillonnés.* Dunod Univertité, Bordas, Paris, France, 1989.

[Sho85] K. Shoemake. Animating rotation with quaternion curves. *Computer Graphics*, 19(3):245–254, July 1985.

[vdPF93] M. van de Panne and E. Fiume. Sensor-actuator networks. *Computer Graphics*, August 1993. Proceedings of SIGGRAPH'93.

[vdPFV92] M. van de Panne, E. Fiume, and Z.G. Vranesic. Control techniques for physically-based animation. In *Third Eurographics Worshop on Animation and Simulation*, Cambridge, England, September 1992.

[WK88] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

A Newton Integration Scheme for Equations of Motion

Let $\vec{x}(t)$ be the current object position, and $\vec{v}(t)$ its speed. Under a set of external forces $\sum \vec{F}$, we used the "Newton" scheme for integrating the equations of motion during a time interval dt:

$$\vec{v}(t+dt) = \vec{v}(t) + \frac{\sum \vec{F}(t)}{m}\, dt \tag{11}$$

$$\vec{x}(t+dt) = \vec{x}(t) + \vec{v}(t)\, dt + \frac{1}{2}\frac{\sum \vec{F}(t)}{m}\, dt^2 = \vec{x}(t) + \frac{1}{2}(\vec{v}(t) + \vec{v}(t+dt))dt \tag{12}$$

Similarly, the second Newton equation of motion for a solid of inertia tensor J and of current orientation matrix $R(t)$ gives:

$$\vec{\omega}(t+dt) = \vec{\omega}(t) + J^{-1}\left(\vec{T}(t) - \vec{\omega}(t) \wedge J\vec{\omega}(t)\right)\, dt \tag{13}$$

$$R(t+dt) = \left(I + \frac{1}{2}(\tilde{\omega}(t) + \tilde{\omega}(t+dt))\, dt\right)\, R(t) \tag{14}$$

where $\tilde{\omega}$ is computed from $\vec{\omega}$ in such a way that: $\forall \vec{a}\ \ \tilde{\omega}\vec{a} = \vec{\omega} \wedge \vec{a}$.

Newton scheme computes an exact solution for constant forces applied during the time interval dt. It produces a better result in practice than Euler integration scheme used in [GG94].

Chapitre 7

Scripting interactive physically-based motions

Ce travail qui apporte un complément indispensable au précédent en permettant la mise au point d'animations "à scénario", qui fait intervenir des contraintes de synchronisation entre divers objets animés par des modèles physiques, fait l'objet d'un papier accepté pour une conférence internationale :

[**LG95**] Scripting Interactive Physically-Based Motions with Relative Paths and Synchronization. Alexis Lamouret et Marie-Paule Gascuel. A paraître dans les actes de Graphics Interface'95. Quebec, Canada, mai 1995.

Résumé

Nous présentons une nouvelle approche pour faciliter l'utilisation des modèles physiques par des animateurs. L'idée maîtresse consiste à laisser l'utilisateur guider le mouvement à un haut niveau de contrôle, en donnant les trajectoires approximatives désirées et des contraintes de synchronisation entre les objets au cours du temps, tandis qu'un module de simulation se charge de calculer le mouvement final des objets. En particulier, l'utilisateur n'a plus a prendre en compte les problèmes comme l'évitement des interpénétrations, les déformations dues aux collisions, ou le réalisme du mouvement. Ces problèmes seront résolus au cours de la simulation, en fonction des propriétés physiques qui ont été spécifiées pour chaque objet.

Isolés ou composants d'une structure articulée complexe, les objets sont guidés par la spécification de positions et d'orientations clés. Si nécessaire, chacune de ces clés peut être définie dans le repère d'un autre objet en mouvement, ce qui facilite la mise au point de mouvements coordonnés complexes. Un scénario est spécifié pour préciser les contraintes de synchronisation entre les mouvements des différents objets au cours du temps. Pendant la phase de simulation, les objets règlent automatiquement leur vitesse pour s'adapter aux contraintes qui leur ont été fixées. En particulier, lorsque les collisions viennent retarder le mouvement, les objets se synchronisent automatiquement sur le plus lent. La méthode est purement simulative et ne requiert pas de module d'optimisation, ce qui permet un calcul interactif des animations.

Scripting Interactive Physically-Based Motions with Relative Paths and Synchronization

Alexis Lamouret
Marie-Paule Gascuel
iMAGIS / IMAG*– INRIA
BP 53, F-38041 Grenoble cedex 09, France
email: Alexis.Lamouret@imag.fr,

February 16, 1995

Abstract

This paper presents a novel approach for facilitating the use of physically based models by animators. The idea is to let the user guide motion at a high level of control by giving approximate desired trajectories and synchronization constraints between the objects over time, while a simulation module computes the final motions, thus enhancing realism. In particular, the user does not have to address problems such as interpenetration avoidance, deformations due to collisions, or realism of motion. The simulator will detect and respond to collisions, correct the trajectories, and compute deformations according to each object's physical properties.

The objects, which are either isolated or components of an articulated structure, are guided through the specification of key-position and orientations. If needed, each of these keys can be defined relative to another moving object, which facilitates the design of complex coordinated motions. The global animation is scripted by specifying a graph of synchronization constraints between objects over time. During the animation, objects automatically regulate their speed in order to meet the constraints, even if one of them is slower, or has been delayed by an unexpected collision. As the method uses a forward simulation over time and doesn't require any optimization process, the entire animation is computed at interactive rates.

Keywords: animation, simulation, motion control, collisions.

1 Introduction

Finding good ways of compromising between simulation and control has became one of the main challenges in current Computer Animation research. Traditional animation systems offer precise control on motions and on deformations, through key-frames or key-shapes, but provide no help at all for improving

*IMAG is a Research institute affiliated with CNRS, Institut National Polytechnique de Grenoble and Université Joseph Fourier.

the realism of the resulting animation. In particular, there is no automatic process for detecting physically infeasible paths, avoiding inter-penetrations, computing deformations during contacts, or suggesting coherent speed variations. Conversly, pure physically-based simulators do not seem adequate to be used for production, due mainly to their lack of control. Generating motions by directly specifying applied forces and torques with time can be sufficient for simulating inanimate objects, but seems impossible when several interacting characters need to be synchronized according to a given script.

Could physically-based simulation become a tool for animators ? Probably yes if we can preserve a familiar interface, while offering the benefits of a simulation for enhancing realism and reducing the animator work.

This paper presents a method for scripting and synchronizing physically-based motions. The animator guides an object by giving a set of successive key-positions and key-orientations, each of them being defined either in world-fixed coordinates or with respect to other moving objects. The final motion is generated during a forward simulation process, thus correcting the predefined path according to the object's physical parameters and to the events such as collisions detected during motion. Speed variations along the paths are computed by the simulation, which ensures their coherence with the path complexity and with the collisions, frictions, and contacts occurring during motion. The user can synchronize different objects (or different components of a complex structure) by specifying a graph of temporal events which links together some of the key-positions defined for each of them. This enables the design of complex scripts that include appointments (the objects will adapt their speed if one of them is delayed), and that control the synchronization of the different motions.

The remainder of this paper is structured as follows: Section 2 briefly reviews related work. Section 3 describes a method, to appear in [LGG95], for introducing some trajectory control in the physically-based animation of a given object. Some extensions to this original method are then presented. The method introduced in Section 4 allows to define some of the key-positions of an object relatively to other moving solids. Section 5 develops an algorithm for adding synchronization constraints to link the motions of different objects. It ensures that specific key-positions on the different objects trajectories will be reached at the same time, even when one of the objects has been delayed by a collision, or by a more complex path. Section 6 explains how to use these temporal constraints for scripting and synchronizing complex motions in a full animation sequence. We conclude and discuss work in progress in Section 7.

2 Related Work

Constraints methods [BB88, Ove91, GG94] and inverse dynamic techniques [IC87, Dum90, Ove91, Ove93] enable the specification of trajectories for some of the components of a complex structure, while leaving the system to animate the other degrees of freedom. They do not offer any help for improving the realism of the specified motions. In particular automatic collision processing (that would possibly include deviations from the desired motion) cannot take place

for the components for which either position or orientation is specified.

Optimization techniques [BN88, Han93, WK88, Coh92] provide a convenient interface for the user, who defines the animation through a set of key-frames. Physical models are used for finding correct interpolations between key-positions during a minimization process (the criteria most frequently used results in the minimization of the amount of energy needed to perform the motion). These methods however either require several forwards and backwards simulations over time, or the use of a global optimization process. Consequently, they do not produce animations at interactive rates. Moreover, most of these methods cannot be associated with automatic collision detection and response: contacts must be predefined by the user as extra constraints holding during specified time intervals.

Methods based on controllers have the advantage of being compatible with forward simulation techniques, which facilitate automatic collision detection and response. Both specialized controllers [RH91, vdPFV92, DLC93] and techniques for automatically generating an adequate controller [vdPF93, NM93] have been previously presented. These approaches are very promising, but they are not aimed at producing complex motions that are synchronized according to a user-defined script.

We have recently proposed a method for using a generalized controller to keep an object close to a desired path, while providing some automatic collision processing [LGG95]. The remainder of this paper first reviews this method and extends it to enable the definition of relative paths. Then, we present a new synchronization algorithm, and describe the way we use it for scripting an animation.

3 Combining Simulation and Trajectory Control

This section explains how to guide the trajectory of a single physically-based object (that can be a component of an articulated structure) while using the simulation for improving the user-defined path. The final trajectory will be smoothed according to the object mass and inertia, corrected with respect to the eventual collisions and contacts detected during motions, while deformations and adequate speed variations of the object will be produced. A more detailed description of this method can be find in [LGG95].

The central idea is to leave the user to define a rough trajectory for the object, either in translation or in orientation (or both), by setting some key-frames. The user also gives a *base distance*, d_{base}, that gives the precision with which the desired trajectory should be followed[1]. The system ensures that any point of the predefined trajectory will be reached with this precision. An actuator, either in translation or in rotation, is associated with the object. It will be used for generating a force (or a torque) pulling the object near the predefined path. The next section explains how we use a generalized Proportional-Derivative controller (PD-controller) for computing the actuator action.

[1]We use the quaternion representation of orientations to enable a simple definition of a "distance" between two of them.

3.1 Basic trajectory control

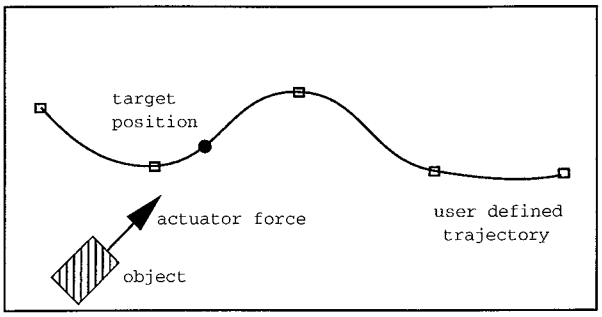


Figure 1: Actuated object guided along a path

Let us first consider an isolated object (with no articulation and that is not submitted to lasting external forces). The method for computing the actuator action is to perform a race between the object and a *target position* following the pre-defined path (see Figure 1). At each animation step:

1. The target moves forwards onto the path, trying to maintain the distance d_{base} with the object.

2. The actuator produces a force computed for covering a fixed portion of the distance to the target ;

3. This force is integrated in the object's equations of motion together with eventual reaction forces due to collisions. The object is moved to its next position.

The benefits of this algorithm are that the system object/target reaches a constant speed (controlled by the user) if nothing occurs, while a collision between the object and an obstacle produces some delay, without introducing any extra lost of precision in the final trajectory. In effect, if the object has been deviated by a collision, the target just waits for it.

We now detail the two first steps of this algorithm:

Computing the target motion

As mentioned before, the desired trajectory followed by the translation or orientation target has been pre-specified by the user through either a set of key-positions or key-orientations. Keys are interpolated by a spline curve (defined in quaternion space for orientations). The target position, an increasing parameter along this curve, is recomputed by binary search in order to maintain the base distance with this object. If it is impossible (the object is too far away), the target just stays at the same location.

Computing the actuator action

Let α be the proportion of the distance between object and target that we want the object to cover during a fixed relaxation time[2] Δt. These two parameters control the ideal speed the system will reach when nothing happens.

The actuator force $\vec{F}$ or torque $\vec{\mathcal{T}}$ enabling the object to perform the desired motion is computed via a generalized version of a PD-controller:

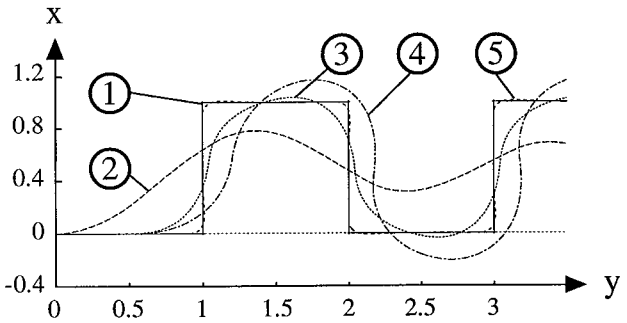
$$\vec{F}(t) = 2m \frac{\alpha\left(\overrightarrow{x_{\text{target}}} - \vec{x}(t)\right)}{\Delta t^2} - 2m\frac{\beta\vec{v}(t)}{\Delta t} \tag{1}$$

$$\vec{\mathcal{T}}(t) = \frac{2.J\left(\alpha\vec{W} - \beta\vec{\omega}(t)\right)}{\Delta t} - \vec{\omega}(t) \wedge J\vec{\omega}(t), \text{ where } \vec{W} = \frac{R_{\text{target}}R_{\text{object}}^{-1} - I}{\Delta t}. \tag{2}$$

were m is the object mass, J its inertia tensor, $\vec{x}$ its position, R its orientation and $\vec{v}$ and ω its linear and rotation speed vectors, respectively.

Detailed comments on these two formulas can be find in [LGG95]. We can stress that the control parameter β sets the smoothness/damping of the motion compared to the given trajectory:

- if $\beta = 0$, the object may oscillate indefinitely from one side to the other of the desired trajectory.
- if $\beta = 1$, there are no more oscillations, but the inertia is not taken into account in the generation of motion.
- In-between, we can tune β to obtain a more or less damped motion (see Figure 2).
- $\beta_{critical} = \sqrt{2\alpha}$ (obtained from the control theory) appears to be a good value for the quickest convergence.



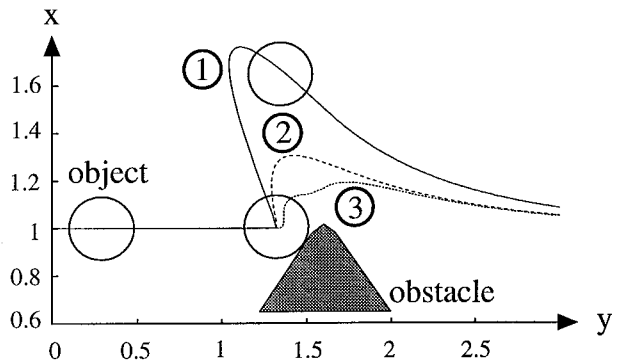
This figure shows various motions for a trajectory physically impossible (curve 1). Changing the base distance (curves 2 and 5) enables to choose how much we want to smooth the trajectory. Changing α within a certain range, and with the same β affects only the speed of the motion. Changing β (curves 2, 3 and 4) affects the damping of motion.

Figure 2: Tuning the actuator and target parameters

[2]We use here a fixed relaxation time because our animation system handles an adaptive time step. The way we compute the actuator action must not be perturbed by time-step changes due to other processes (such as the collision processing module).

3.2 Articulated objects / Lasting external forces

The actuator force (or torque) described above is sufficient to deal with external forces applied only during a few time-steps, such as response to collision. Figure 3 shows how an object comes back near the desired path after a deviation due to an obstacle (our system uses the algorithm of [Gas93] for detecting and responding to collisions).



In this figure an actuated object collides with a rigid obstacle. The trajectory given by the animator is a simple horizontal line passing through the obstacle. We see various behaviors depending on the mass (1 and 2) and the stiffness of the object (1 and 3). In (3) the object deforms and slides around the obstacle, while in (1) and (2) it bounces more or less violently.

Figure 3: Deviation from the user-defined path due to a collision

However, the effect of lasting forces such as weight or connection forces between neighboring components in an articulated structure should not affect the precision with which the desired path is followed. To deal with these lasting external forces, we add a correction term to the force or torque computed in equations (1) or (2). This term is computed from an estimation of the sum of external actions applied on the object during the previous time-step (we observe the difference between expected position and reached position during the last time interval). We use a filter which smoothes the corrections terms over time, and produces the delay needed for maintaining an adequate deviation in the case of a sudden external action.

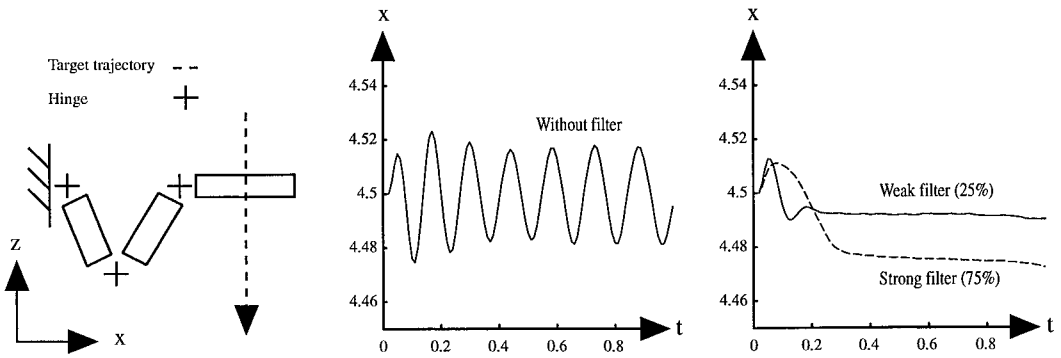


Figure 4: Filter effect on an articulated object.
(a) motion without filtering the actuator forces. (b) motion with a filter.

Detailed equations for computing correction terms and filtering them are given in [LGG95]. Figure 4 shows the results we obtain for a three-link ar-

ticulated arm whose extremity is expected to move down on a vertical line (the method we use for animating articulated bodies is described in [GG94]). Filtering avoids the horizontal oscillations due to the action of neighboring components during the downward motion.

4 Defining Paths relative to another object

Specifying motion relative to other objects may be more useful for some applications than giving a desired location in world fixed coordinates. For instance, if two characters must shake hands, the precise location where they do it is not really important. This section proposes a first contribution to our original method which enables the user to define each key-positions or orientations in local coordinates with respect to another moving object.

The principle is the following. When the user specifies the set of keys defining the desired path for an actuated object, he associates each of them with either the world fixed coordinate system or with another object's local coordinate system.

The algorithm for computing the target motion (see Section 3.1) is modified as follows:

1. We select the control points (the keys) influencing the current target parameter on the spline curve (in practice, we use cubic splines to interpolate both positions and orientations, so each curve segment depends on four control points). We add the control point that immediately follows, as the target may pass through a joint.

2. We convert the coordinates of these control points (if needed) into world fixed coordinates, taking the current locations and orientations of local coordinate systems into account.

   For most applications, the trajectory should not depend on the motion of a control point the target has already passed through, so a control point is "pined" to its position in world-fixed coordinates as soon as the target reaches it (see Figure 5).

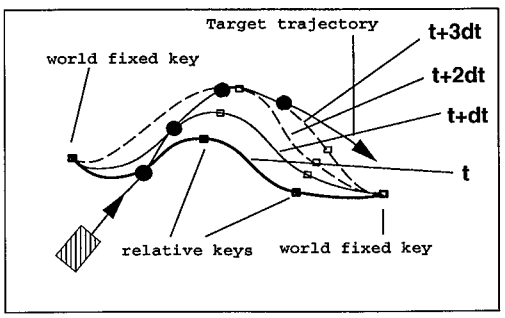


Figure 5: Target motion for a relative path

3. Then, we have the desired spline segments in world fixed coordinates, and we use the usual binary search method for finding the new target parameter.

Since each object does not move too much during a single animation step (this condition is required for any system that performs step by step animation), the target trajectory stays smooth.

Results

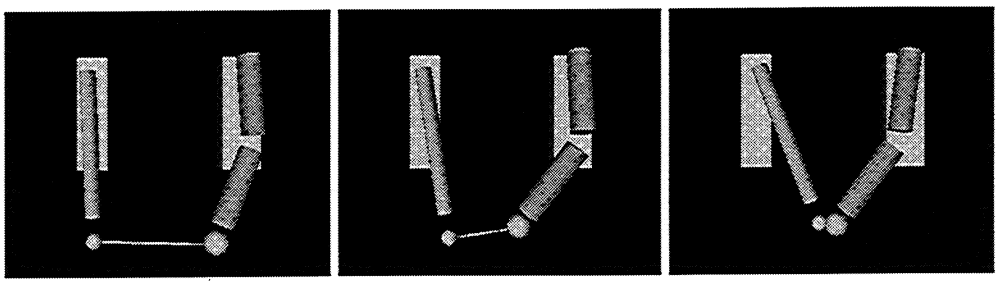


Figure 6: Two articulated arms shaking hands. The final position for each hand is defined in local coordinates relative to the other hand. The location where the shaking takes place is computed by the simulation, and depends on the strengh, mass, intertia and articulations of each arm.

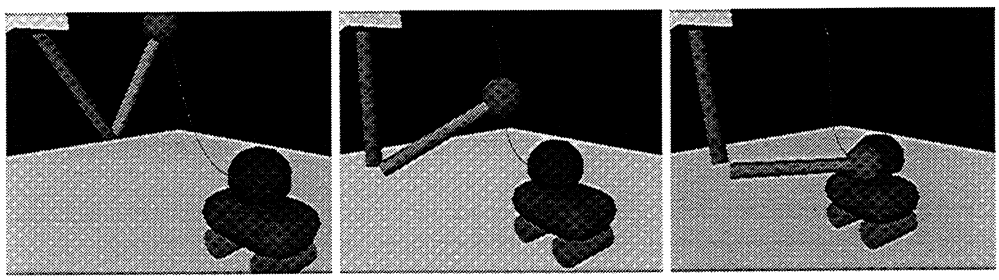


Figure 7: A three links articulated arm has to kick down a ball lying on a moving wagon. The desired final trajectory for the hand (position to reach and tangent to the spline curve) are defined in local coordinates relative to the ball. The contact between the ball and the wagon and the collision with the hand are automatically detected and treated by the simulation.

5 Synchronizing motions

In the method described up to now, we have chosen to compute the timing of motion during the simulation process rather than pre-specifying it. Indeed, the speed variations must depend on the path complexity, and on the events (collisions, contacts) occurring during motion. However, most animation sequences produced in Computer Graphics require careful motion synchronization between the different objects of the scene, or even between the different components of

an articulated character. For instance, modeling walking requires carefull synchronization of legs and arms motions.

Our second contribution to the original method consists in adding synchronization constraints which link the respective motions of various actuated objects. This section defines the method we use for fulfilling a single synchronization constraint between any number of objects. The application to scripting an animation sequence will be presented next.

A synchronization constraint is defined by selecting a set of positions (or orientations) on the different user-desired trajectories of the objects, and imposing the condition that all these positions will be reached at the same time by the object's respective targets. For instance, an appointment between different moving characters in a specific space location can be modeled this way. Another example is a character that must bend his knees while raising his arms. The synchronization is then realized between keys some of which are in translation, and others in rotation.

Performing motions under a synchronization constraint is done by regulating the average speed of the different targets in an auto-adaptive way.

Regulating Targets Speeds

The easiest way to synchronize a set of objects is to regulate all the speeds according to the slowest one. Here, targets are to be synchronized (constraints are defined between target positions), so the the basic idea is to adjust target speeds. As the motion of the target depends upon the associated object, the slowest target will correspond to the slowest object, and we will definitely be able to slow down the others, while it would be uncertain to try to accelerate the slowest.

Another point is that we want the target to recover its original ideal speed after the synchronization has been achieved (targets that have been slowed down should not stay slow, otherwise the current speed of each object would depend of every perturbation occurred since the beginning of the animation!). So, when no reduction of a target speed has been needed during several consecutive time intervals, or when the synchronization constraint has been reached, we increase the speed again.

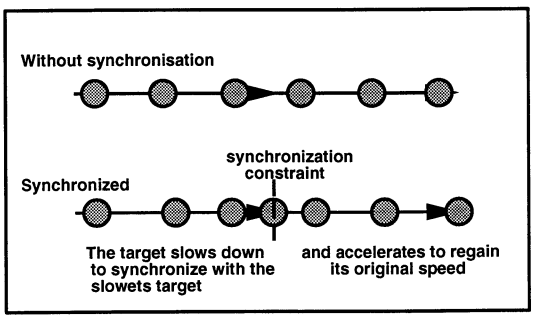


Figure 8: Target Speed Variations due to synchronization

Choice of the parameter to control: If there is no external event (no collision, etc), the average speed of an object, and hence of its target, is $\alpha.d_{base}/\Delta t$ (see Section 3.1). To modify the speed of a target, we must change one of these parameters. Moreover, apart from the different speeds of objects relative to each other, we don't want to change any characteristic of the movement: neither the damping of the trajectory (based on the coefficient α/β^2), nor the precision with which the object will follow it (given by d_{base}). Modifications of the target speed will thus be achieved by regulating α.

Synchronization algorithm: The algorithm is based upon the spline parameter. We have different actuated objects that each try to reach a specific *goal position* on their trajectory at the same time. To evaluate the time each object will need to reach its *goal position*, we have to find a criterion. The simple metric distance will not help, since the trajectory may be more or less complex depending on the considered object. We use instead the length of the trajectory curve, and compare the distance the target covered in one time step to the total distance from the current point to the point of the appointment.

In our implementation, we get direct computation of the *curvilign abscissa* s (defined as the length of the spline between the first and the current point) by using reparametrized splines curves, for which the spline parameter equals an approximation of the absissa[3].

Thus the time for a target τ to reach its goal is estimated as:

$t_{estimated}(\tau) = dt.(s_{goal}(\tau) - s_{current}(\tau))/ds(\tau)$, where ds is the length covered in the previous timestep dt.

At each time step, the module that regulates the target speed operates as follows:

- Check if each target has reduced its speed recently. If not, it is allowed to accelerate by a certain percentage of its current speed.
- Evaluate the time each target would spend at its current speed to reach its desired goal, and keep the maximum as the desired time:

$$t_{desired} = max_{\tau}(t_{estimated}(\tau))$$

- Reduce each target speed so that it will reach its goal in a time $t_{desired}$: $\alpha_{new}(\tau) = (t_{estimated}(\tau)/t_{desired}).\alpha_{old}(\tau)$.

Then we apply the method of Section 3.1 to move targets, compute actuators forces or torques and simulate all the objects.

[3]These reparametrized splines are described in [Lyo94], and are computed by storing an array of values $ds = \sqrt{(dx/du)^2 + (dy/du)^2 + (dz/du)^2}du$ (the new parametrization) between sample points distributed along the curve.

Example

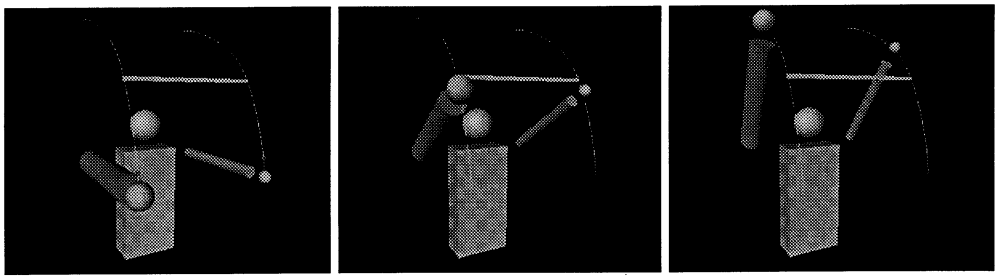


Figure 9: Two articulated arms, with different weights and intertias, must fullfill a synchronisation constraint while going up: the intermediate key-positions are to be met at the same time. The results shows how the strongest arm descelerates to wait for the slowest, and how it comes back to its ideal speed after the appointement.

6 Scripting an Animation Sequence

The method we have just presented for synchronizing motions can be used in a larger scale, in order to script an entire animation sequence.

In the previous section, we demonstrated how to organize coordination of several targets for a single synchronization constraint. More generally, if the user is animating n actuated objects, he/she may need to define several synchronization events to occur over time. Each of these events may link together several of the targets, and some succession order may need to be specified between some of them, but not necessarily between all.

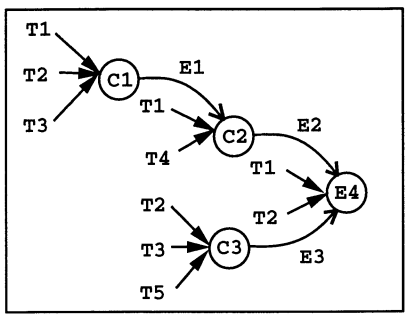


Figure 10: The graph of synchronization constraints

The C_i ($i = 1..3$) represent the constraints, the T_j ($j = 1..5$) are the targets involved for each of them, and the E_k ($i = 1..3$) are the oriented edges, defined the user, which specified the desired sequencement between events (an edge from C_i to C_j tells that the synchronization constraint C_i must take place before C_j.

The sequencing of synchronization constraints can then be defined by the user through a Petri network, as depicted in Figure 10. This network defines descendance relationship between some of the constraints. At a given time, only a limited number of these constraints are activated, i.e. only those whose

parents have already been reached (for instance constraint C_4 in the figure will be activated only when C_2 and C_3 have been reached).

To implement this process, we store the synchronization constraints in a structure where each of them successively reaches three states:

- *Waiting*, which means that no effort is made to take this constraint into account, so targets linked only to it are not constrained yet.
- *Active*, which means that targets are trying to synchronize according to this constraint.
- *Done*, which means nothing is done anymore, so that the targets depending only on it return to their original state (unconstrained).

A "waiting" constraint becomes "active" as soon as all its parents are "done".

Other algorithms can be chosen for activating constraints. For instance, it may be usefull to take them into account earlier, by specifying that they will be considered as soon as a specified number of their parents are done.

Results

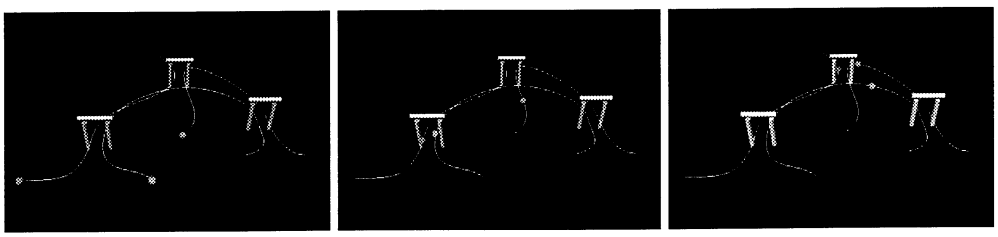


Figure 11: Three spheres representing simplified characters are synchronizing themselves according to a user-defined script: couples of spheres must pass together under each of the three archs. The automatic speed regulation makes the spheres accelerate or descelerate according to the next constraint and to the others motion.

7 Conclusion

Designing an appealing key-frame animation requires a great amount of specialized knowledge from the animator, who spends a lot of time specifying motions and deformations. This very precise control is certainly desirable for animating the foreground, but reducing the time the animator spends on secondary motions would be desirable.

In this paper a method was presented for scripting secondary motions with a high level of control. The user gives some key-positions and key-orientations for those of the objects (or components of an articulated structure) whose trajectory needs to be guided. These keys are either defined in world-fixed coordinates or relative to other objects. Spending a lot of time adjusting them is not necessary: inter-penetration avoidance (with subsequent deviation from the predefined path), deformations due to contacts, and adequate speed variations

are automatically generated by the simulation. In addition, the key-features defined by the user can be linked together by temporal constraints. This enables the specification of a script that controls synchronization between the various elementary motions. For instance, appointments can be achieved even in a scene where objects are delayed by unexpected collisions.

The method consists of associating the objects with actuators capable of pulling them towards a target position that follows the desired path. The target auto-regulates its speed according to the object motion and to the synchronization constraints to be met. As the algorithm works during a single forward simulation over time, animations are generated at interactive rates. The resulting motion is as realistic or as unrealistic as the animator wants since it stays close to the predefined path, but not exactly on it if the path can be improved.

Work in progress includes attempts to combine our approach with a technique for computing muscle action for character animation. In this case, using muscles for generating torques at hinges as much as possible would be better than simply pulling objects as marionnettes. Unfortunately, the exclusive use of muscles and reaction forces is an unsolved problem for complex aperiodic motions. We think that combining actuators with muscles would lead to an easier solution in this case.

Acknowledgements

Many thanks to Michiel Van de Panne for very interesting discussions, to Jean-Dominique Gascuel for his assistance, and to Georges Drettakis for carefully re-reading this paper.

References

[BB88] R. Barzel and A. Barr. A modeling system based on dynamic constraints. *Computer Graphics*, 22(4):179–188, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[BN88] L. Shapiro Brotman and A. N. Netravali. Motion interpolation by optimal control. *Computer Graphics*, 22(4):309–315, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

[Coh92] M. Cohen. Interactive spacetime control for animation. *Computer Graphics*, 26(2):293–302, July 1992. Proceedings of SIGGRAPH'92 (Chicago, Illinois, July 1992).

[DLC93] Y. Delnondedieu, A. Luciani, and C. Cadoz. Physical elementary component for modeling the sensori-motricity: the primary muscle. *Fourth Eurographics Animation and Simulation Workshop*, September 1993.

[Dum90] G. Dumont. Animation de scènes tridimensionnelles : la mécanique des solides comme modèle de synthèse du mouvement. *Thèse de Doctorat*, Université de Rennes I, May 1990.

[Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics*, pages 313–320, August 1993. Proceedings of SIGGRAPH'93 (Anaheim, California, August 1993).

[GG94] M.P. Gascuel and J.D. Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer*, March 1994. A first version of this paper appeared in the *Third Eurographics Workshop on Animation and Simulation*, Cambridge, UK, Sept 92.

[Han93] Gabriel Hanotaux. Techniques de contrôle du mouvement pour l'animation. Thèse de doctorat, École Nationale Supérieure des Mines de Saint-Étienne, Université de Saint-Étienne, April 1993.

[IC87] P.M. Isaacs and M.F. Cohen. Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics. *Computer Graphics*, 21(4):215–224, July 1987. Proceedings of SIGGRAPH'87 (Anaheim, California, July 1987).

[LGG95] Alexis Lamouret, Marie-Paule Gascuel, and Jean-Dominique Gascuel. Combining physically-based simulation of colliding objects with trajectory control. *To appear in The Journal of Visualization and Computer Animation*, 1995.

[Lyo94] Ch. Lyon. Introduction de l'animation descriptive dans un contexte d'animation sous contraintes. *Rapport de stage*, DEA d'Informatique de l'ENSIMAG/UJF, September 1994.

[NM93] J.T Ngo and J. Marks. Spacetime constraints revisited. *Computer Graphics*, August 1993. Proceedings of SIGGRAPH'93.

[Ove91] C. Van Overveld. An iterative approach to dynamic simulation of 3-D rigid-body motions for real-time interactive computer animation. *The Visual Computer*, 7:29–38, 1991.

[Ove93] C. Van Overveld. Building blocks for goal-directed motion. *The Journal of Visualization and Computer Animation*, 4:233–250, 1993.

[RH91] M. Raibert and J. Hodgins. Animation of dynamic legged locomotion. *Computer Graphics*, 25(4):349–358, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

[vdPF93] M. van de Panne and E. Fiume. Sensor-actuator networks. *Computer Graphics*, August 1993. Proceedings of SIGGRAPH'93.

[vdPFV92] M. van de Panne, E. Fiume, and Z.G. Vranesic. Control techniques for physically-based animation. In *Third Eurographics Worshop on Animation and Simulation*, Cambridge, England, September 1992.

[WK88] A. Witkin and M. Kass. Spacetime constraints. *Computer Graphics*, 22(4):159–168, August 1988. Proceedings of SIGGRAPH'88 (Atlanta, August 1988).

Chapitre 8

Implicit surfaces for medical organs reconstruction

Ces travaux décrivent l'application que nous avons développée pour la reconstruction semi-automatique d'organes à l'aide de surfaces implicites. Le papier ci-joint vient d'être accepté pour publication :

[**TBG95**] Implicit Surfaces for Semi-Automatic Medical Organs Reconstruction. Nicolas Tsingos, Eric Bittar et Marie-Paule Gascuel. A paraître dans les actes de Computer Graphics International'95. Leeds, Grande Bretagne, juin 1995.

Résumé

Cet article présente une méthode destinée à la reconstruction d'organes, pour laquelle les données de départ sont des nuages de points bruités, répartis non uniformément, et dont les vecteurs normaux ne sont pas connus. La topologie et la géométrie de l'organe à reconstruire peuvent être quelconques, comme dans le cas d'une vertèbre qui comporte un trou et plusieurs embranchements.

La méthode proposée utilise des surfaces implicites engendrées par des squelettes. Le principe est identique à celui de [Mur91] : il s'agit de minimiser une énergie représentant la distance entre l'ensemble de points et l'iso-surface, les squelettes utilisés se subdivisant automatiquement si nécessaire. Nous introduisons cependant plusieurs amélorations, qui permettent de réduire les temps de calcul de manière significative. En premier lieu, nous utilisons des fonctions potentiel locales et peu coûteuses. Cela nous permet de définir un critère local pour la subdivision des squelettes, qui nous indique immédiatement quel est le prochain à subdiviser. Une autre optimisation consiste à découper l'espace des données en plusieurs "fenêtres de reconstruction", où seules les données locales sont prises en compte. Enfin, nous permettons à l'utilisateur de tirer avantage de ses connaissances initiales sur la surface grâce à un procédé de reconstruction semi-automatique : une interface interactive est fournie pour visualiser les données, positionner quelques squelettes de départ, définir les fenêtres de reconstruction, et afficher les résultats intermédiaires. Des exemples tirés ou non du champs d'application médicales sont présentés.

Implicit Surfaces for Semi-Automatic Medical Organs Reconstruction

Nicolas Tsingos † *, Eric Bittar ‡, Marie-Paule Gascuel †,

† iMAGIS †/ IMAG
BP 53, F-38041 Grenoble cedex 09, France
Nicolas.Tsingos@imag.fr, Marie-Paule.Gascuel@imag.fr

‡ TIMB - TIMC / IMAG
Institut Albert Bonniot, Faculté de Médecine de Grenoble, 38 706 La Tronche, France
Eric.Bittar@imag.fr

Abstract

This paper proposes a new method based on implicit surfaces for medical organs reconstruction. The original data are noisy scattered points of an organ, that may be arranged in any non-uniform repartition. The knowledge of the normal vectors is not needed. Organs of any geometry and topology can be reconstructed, as for instance a vertebra that is characterized by a hole and by several branchings.

The proposed method uses implicit iso-surfaces generated by skeletons, that are a particularly compact way of giving a smooth representation of a surface. Validity of the reconstructed shape is insured, since implicit surfaces bound a well defined volume. As in a previous work [Mur91] the principle is the minimization of an energy that represents the distance between the set of points and the iso-surface. Skeletons generating field functions are automatically subdivided when needed. But the algorithm we use is novel. Firstly, we introduce local control on the reconstructed shape thanks to local and less expensive field functions. Secondly, we propose a much more efficient skeleton subdivision process, based on a notion of "skeleton energy" that gives a robust criterion for choosing the next skeleton to subdivide. Another optimization consists in splitting the data space into several overlapping windows, where only local data points are used for reconstruction. Implemented as a semi-automatic process, the reconstruction module enables the user to take benefits of his initial knowledge of the surface to guide computations. We use an interactive system for visualizing the data, initially positioning some skeletons, defining local reconstruction areas, and visualizing intermediate results. We have successfully applied the method to both non-medical and medical data.

Keywords: Implicit surfaces, medical applications, shape reconstruction

*currently at ICP/INPG-Université Stendhal, Institut de la Communication Parlée, B.P. 25 X, F-38041 GRENOBLE CEDEX 09, FRANCE

†iMAGIS is a joint project of CNRS, INRIA, Institut National Polytechnique de Grenoble and Université Joseph Fourier.

1 Introduction

The problem of shape reconstruction of an object is a center of interest of many researchers in various fields, such as medical applications, computer vision, computer animation, or CAD. The medical community needs to reconstruct the human organs for diagnostics, as well as for surgery planning or simulation. The information can be contained into images, volumes, or can merely be a set of scattered points. Either coming from a range sensor or being the result of images segmentation, these points are the basis of most of the reconstruction algorithm. The large interest for the problem explains why so many different approaches have been developed:

Boissonnat [Boi84, BG92] proposes to use the Delaunay triangulation of the set of points. Edelsbrunner and Mücke [EM92] introduce a parameter α that enables to tune the degree of detail while deriving a shape from the Delaunay Triangulation. Attali [ABM94] computes the Voronoï graph of the points to build the "skeleton" of the object and reconstruct its shape. An other approach, introduced in [HDDW92] and completed in [HDDH94] by a mesh optimization, consists in using graph traversal techniques to calculate a signed distance function from the points, and build an iso-surface from this function.

A different philosophy consists in deforming a surface or a volume to fit the data points. The introduction of energy-minimizing curves, known as "snakes", is due to Kass, Witkin and Terzopoulos [KWT88]. The idea was then widely used and developed [TWK88], and the active contour turned to an active surface, deformed by internal forces, such as pressure, and by external forces based on the data to fit [CC92]. The implementation relies most of the time on splines, but Miller also proposes a Geometrically Deformed Model based on triangles [MBE+91]. Another option is to use deformable implicit volumes. Sclaroff and Pentland [SS91] fit a super-quadric by use of modal deformations and displacement maps. Terzopoulos and Metaxas choose a physically based approach for local and global deformations of super-quadrics [TM91].

Most of the developed methods only reconstruct surfaces of fixed topological type. However, a few papers describe models that are able to change their topology: Delingette [Del94] introduces the "Simplex Meshes" for which the user can interactively adapt topology. Leitner uses an adaptative spline surface [Lei93], and Lachaud and Bainville a model derived from triangulation [LB94].

Muraki [Mur91] is the first one who uses implicit iso-surfaces generated by skeletons for shape reconstruction. An initial skeleton is positioned at the center of mass of the data points, and divided until the process reaches a correct approximation level. But Muraki does not take a full advantage of the potentialities of implicit surfaces, and the reconstruction process he proposes is very slow.

This paper presents a new method for reconstruction with implicit surfaces generated by skeletons. We introduce local control on the reconstructed shape thanks to a local field function, that enables the definition of local energy terms associated with each skeleton. This leads to a much more efficient skeleton subdivision process, since we get a robust criterion telling which skeleton should be

divided next. Unlike in Muraki's work, the knowledge of the normal vectors at the data points is not needed. The method works as a semi-automatic process: the user can visualize the data, initially position some skeletons thanks to an interactive implicit surfaces editor, and further optimize the process by specifying several "reconstruction windows", that slightly overlap, and where surface reconstruction follows a local criterion. If needed, different precisions of reconstruction can be defined in each window. The shapes to reconstruct can be of any topology and geometry, and for instance include holes and branchings. We show reconstruction experiments from noisy medical data, for which scattered points are arranged in non-uniform repartition.

The remainder of this paper develops as follows: Section 2 introduces implicit surfaces generated by skeletons and discusses Muraki's reconstruction method. Section 3 details the amelioration we propose to the original algorithm, that offer both local control on the reconstructed shape and a robust heuristic for choosing the skeleton to split. The user interface we offers for semi-automatic reconstruction is described in Section 4, and results are presented in Section 5. Section 6 concludes and discusses work in progress.

2 Shape reconstruction with implicit surfaces

2.1 Implicit surfaces generated by skeletons

Developed up to now as a tool for free form modeling [WMW86, WW89, BW90, BS91], implicit surfaces generated by skeletons constitute a good alternative to traditional implicit surfaces directly given by an analytical equation. Indeed, manipulating simple geometric primitives such as skeletons facilitates the design of complex shapes.

An implicit surface S generated by a set of skeletons $S_i (i = 1..n)$ with associated "field functions" f_i is defined by:

$$S = \{P \in I\!R^3 \ / \ f(P) = iso\} \text{ where } f(P) = \sum_{i=1}^{n} f_i(P)$$

- iso is a given scalar iso-value.
- The skeletons S_i can be any geometric primitive admitting a well defined distance function.
- The field functions f_i are monotonically decreasing functions of the distance to the associated skeleton. They can be defined, for instance, by exponential functions [Bli82], by pieces of polynomials [NHK+85, WMW86], or by parametrized procedural functions [KAW91].

The surface surrounds a volume defined by $f(P) \geq iso$, and normal vectors are directed along the gradient of f.

2.2 Muraki's reconstruction method

Muraki [Mur91] presents an automatic method for generating an implicit shape description from range data that include both points and normal vectors.

The surface used for reconstruction is defined by Blinn's "blobby model" [Bli82]. Only point-skeletons are used, and fields are expressed as exponential functions of the distance $d(P, S_i)$:

$$f_i(P) = b_i\, e^{-a_i d(P,S_i)}$$

where b_i can be either positive of negative[1].

The principle of the reconstruction algorithm consists in minimizing an energy that represents the "distance" between the current implicit surface and the data. Muraki proposes the use of three terms:

- E_{value} fits the surface to the data points:

$$E_{\text{value}} = \sum_{j=1}^{M} (f(P_j) - iso)^2$$

- E_{normal} uses normal vectors to insure that the external side of the iso-surface corresponds to the external side of the object:

$$E_{\text{normal}} = \sum_{j=1}^{M} \frac{|nj - N(Pi)|}{||N(Pi)||^2}$$

- E_{shrink} expresses constraints on the relative values of the parameters a_i and b_i of the n skeletons that avoid surface degeneration in low-sampled areas:

$$E_{\text{shrink}} = (\sum_{i=1}^{n} {a_i}^{-3/2} |bi|)^2$$

The total energy E is given by the expression:

$$E = \frac{1}{M} (E_{\text{value}} + \alpha\, E_{\text{normal}}) + \beta\, E_{\text{shrink}}$$

where M is the number of data points.

Directly using n skeletons to reconstruct the surface would be a very computer intensive process, since it would require solving a non-linear minimization problem with $5n$ unknowns (the 5 parameters a_i, b_i, x_i, y_i, z_i for each skeleton). Muraki performs instead a progressive refinement process. The main steps of the algorithm are:

1. Position a single skeleton at the barycenter of the data points.
2. Make trials, independently for each skeleton S_i:
   - Split S_i into two skeletons S'_i and $S"_i$:
   - Minimize energy by modifying only S'_i and $S"_i$'s parameters ;
   - Store the final energy.
3. Keep the modification of step 2 that gave the lowest energy (all the other trials are not used anymore).
4. If the energy is sufficiently low, stop the process ; else, go back to step 2.

[1] Using a "negative skeleton" ie. a skeleton with a negative field reduces the implicit volume, while positive skeletons extend it.

2.3 Discussion

Using implicit surfaces generated by skeletons for reconstructing smooth shapes seems to be a very good idea, since these surfaces are particularly convenient for defining an energy that gives a current "distance" to the data, and since they can be stored in a very compact way (position and field parameters for each skeleton).

However, several problems can be identified, that seem to forbid the direct use of Muraki's method for our goal of medical organs reconstruction:

- The choice of an exponential field function doesn't give any locality on the reconstructed shape. Every data point is influenced by all the skeletons, even those that are distant. In consequence, each refinement changes the shape everywhere, even in areas where the reconstruction was already sufficient.

- The initialization is performed with a single skeleton, positioned at the barycenter of the data-points. This gives quite good results for reconstructing heads (the only examples given in the paper), for which the progressive refinement of an implicit sphere is well adapted. But one can wonder how the process would work for arbitrary shapes, with holes and branchings, such as a vertebra. In particular, the initial skeleton could be located inside a hole, probably perturbating the reconstruction.

- The very important computational time is really limitative: the given examples, that are said to take "a few days", correspond to 2893 data points. Direct use of the method on the echography of an organ (about 25000 scattered points) would be intractable.

  The most expensive process seems to be the way the next skeleton to split is selected. The author admits that trying every skeleton for just keeping the solution that decreases the energy the most is very slow, and should be changed. In practice, some of the examples in [Mur91] are computed by merely splitting each skeleton successively, in the arbitrary order given by the current list of skeletons. As this method doesn't seem very satisfactory as well, better heuristics need to be found.

3 A novel method for reconstruction

This paper proposes a more efficient method for shape reconstruction with implicit surfaces, that we are experimenting on medical data. This section describes our main contributions for improving Muraki's technique. The main idea we are developing is to take benefits of locally defined field functions to optimize the reconstruction process. This is done by defining a new energy criterion telling which skeleton must be splitted next, and by offering the possibility of processing reconstruction in a more local way, through the use of several "reconstruction windows".

3.1 Choice of a local field function

Local field function, that become zero with their derivatives at a certain radius of influence R have been introduced in the literature for optimizing field computations during the design of complex objects [WMW86]. Another benefits is that these fields offer local control on the implicit surface, which is particularly important for our reconstruction process: splitting and optimizing skeletons in an area should not modify a good reconstruction of the data already obtained in another area.

Contrary to Muraki that uses both positive and negative field functions, we want to reconstruct objects with positive fields only. Indeed, for medical applications that may include a future physically-based simulation of organs deformations, the exclusive use of positive skeletons seems more appropriate.

The chosen field function will be used in a computer-intensive minimization process. So we need a model that is both controlled by very few parameters (in order to limit the dimension of the search space), and that provides efficient computations. In consequence, we are currently experimenting field contributions f_i that are composed of a linear and a quadratic curve segments, and that are parametrized by two parameters (see Figure 1):

- The radius e_i of the sphere created by a skeleton S_i alone (e_i is characterized by $f_i(e_i) = iso$) ;
- The stiffness k_i in e_i, that defines the blending properties of the surface.

The function is expressed by:

$$\begin{aligned} f_i(P) &= -k_i r + k_i e_i + 1 && \text{if } r \in [0, e_i] \\ f_i(P) &= \frac{k_i e_i (e_i - R_i) + 3e_i - R_i}{(e_i - R_i)^3}(r - R_i)^2 && \text{if } r \in [e_i, R_i] \\ f_i(P) &= 0 && \text{elsewhere} \end{aligned}$$

Where $r = d(P, S_i)$ and $R_i = (e_i - \frac{2}{k_i})$ is the radius of influence.

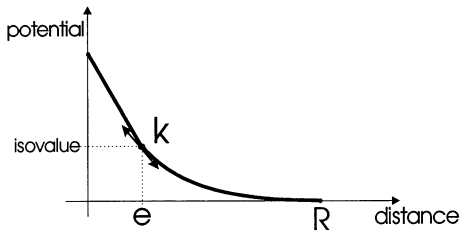


Figure 1: Local field function with two parameters.

As in Muraki's work, there are 5 parameters to optimize for each skeleton: e_i, k_i, x_i, y_i, z_i.

3.2 Basics for semi-automatic reconstruction

The idea is to keep the main principles described in Section 2.2, but with several modifications.

First of all, most medical data do not include any description of normal vectors, that were used by Muraki to insure that the internal volume was reconstructed. However, we are defining a semi-automatic reconstruction process, where the user takes benefits of his knowledge by interactively positioning the initial skeletons with respect to the data. If initial skeletons are positioned inside the volume to reconstruct, the method will work without the need of normal vectors.

In consequence, we redefine the energy to minimize by:

$$E = \frac{1}{M}\left(\sum_{j=1}^{M}(f(P_j) - iso)^2\right) + \frac{1}{n}\left(\alpha_1 \sum_{i=1}^{n} e^{-\beta_1 e_i} + \alpha_2 \sum_{i=1}^{n} e^{-\beta_2 k_i}\right)$$

The two last terms are basically equivalent to E_{shrink} in Muraki's method: they prevent the skeletons from degenerating to negative radius or stiffness (n is the current number of skeletons).

As stressed in Section 2.2, the main problem with Muraki's method is the very intensive computation process, due to the method used for selecting the skeleton to divide next: every solution is tested (which requires running the full minimization) and only the best one is used. The next section describes the heuristic we have developed for accelerating this process.

3.3 An efficient subdivision algorithm

Selecting the skeleton to divide can be done without any extra minimization, by using the new local properties of the field functions. Indeed, the best candidate for splitting is the skeleton whose area of influence corresponds to the zone of the surface were the reconstruction is the worst. See Figure 2.

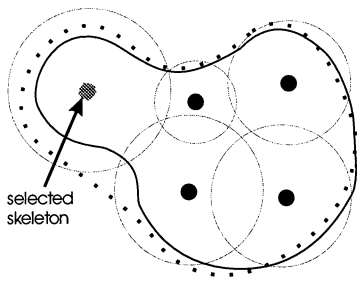


Figure 2: Selection of the next skeleton to subdivide

So we define the splitting criteria as:

$$C_i = \frac{1}{m_i}\left(\sum_{j=1}^{m_i}(f(P_j) - iso)^2\right)$$

where the sum is made only on the m_i point that are inside S_i's sphere of influence.

At each iteration of the algorithm, the skeleton with greatest C_i is replaced by two skeletons, whose 10 parameters are first optimized while keeping the

other skeletons and fields constant. Then, we process a few steps of global optimization before going to the next skeleton subdivision (this global optimization step reduces the final number of skeletons: in some cases, just adjusting all the current skeletons at the same time all is sufficient for decreasing the energy to the desired threshold).

3.4 Reconstruction windows

For large scattered points databases, using all the points for every step of energy minimization can be really intractable, since the energy includes a sum of field terms over the points. On an other hand, many objects to reconstruct are easy to decompose into several specific zones, called "reconstruction windows", where reconstruction can be processed separately. For instance, 3 different windows can be defined for the object in Figure 3. The energy to minimize in a 3D window will be calculated by only using the data-points located in this window, and the global optimization step after a refinement will only be performed on the skeletons affected to this window. Processing local reconstructions is justified by our use of local field functions: when we move a skeleton, only local points are affected, so recomputing the entire energy is not really useful. Local windows optimize computations since less points are considered in energy terms, and minimization is performed in a space of smaller dimension.

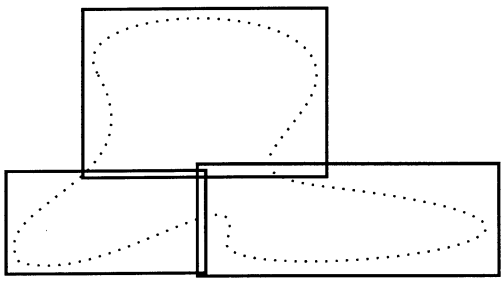


Figure 3: Defining several windows for local shape reconstruction

The algorithm we use for reconstructing data when several reconstruction windows are defined develops as follows. At each reconstruction step:

1. Select the window were:

$$W_k = 1/w_k \, (\sum_{j=1}^{w_k} (f(P_j) - iso)^2$$

   is the highest (w_k is the number of points in this window).

2. Use the criterion of Section 3.3 for selecting a skeleton to divide in this window.

3. Split the skeleton and optimize its two "children", everything else being kept constant. The energy to minimize is computed from the w_k points

and the n_k skeletons of the window:

$$E_k = \frac{1}{m_k}\left(\sum_{j=1}^{m_k}(f(P_j) - iso)^2\right) + \frac{1}{n_k}\left(\alpha_1 \sum_{i=1}^{n_k} e^{-\beta_1 e_i} + \alpha_2 \sum_{i=1}^{n_k} e^{-\beta_2 k_i}\right)$$

4. Process a few steps of optimization in the window in order to further decrease E_k (and W_k).

5. If the global value defined by: $W_{\text{global}} = \sum W_i$ is smaller than the desired threshold, stop the process. Else, go back to step 1.

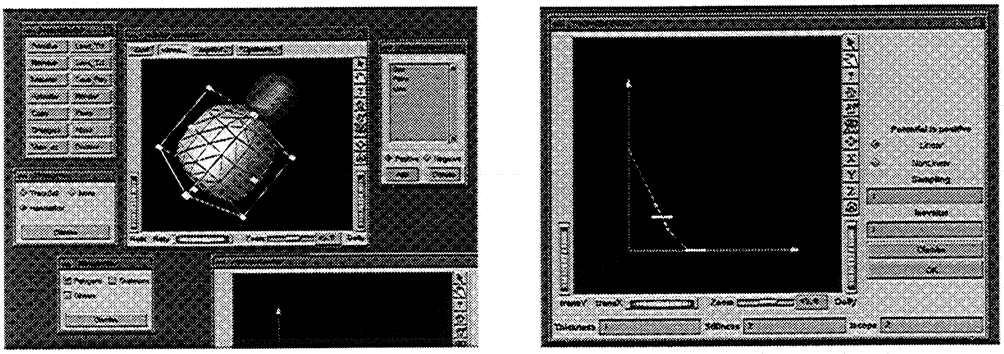
In practice, the list of skeletons and points included into each window are maintained to accelerate the computations.

The user is provided with an interactive interface for visualizing the data, defining reconstruction windows and positioning some initial skeletons. We describe it in the next section.

4 Interface for Semi-Automatic reconstruction

The semi-automatic reconstruction process we have presented is implemented within the system *Fabule*, a platform for interactive modeling and animation that we are developing at *iMAGIS* [Gas94]. It includes an object-oriented library, implemented in C++ and built on *Open-Inventor*, and an interface with the interpreted language *Tcl* and *OSF-Motif* for the easy specification of demonstrators and interfaces.

The user visualizes the data and creates reconstruction windows thanks to 3D widgets provided by *Inventor*. He also interactively specifies skeletons and associated field functions that will initialize reconstruction. Intermediate reconstruction results are visualized within the same interactive system.



(a) General view of the interface (b) Field function editor

Figure 4: Interface for editing skeletons and field functions

We offer an interactive graphic interface for modeling skeletons, visualizing implicit surfaces, and editing the field function parameters (see Figure 4). Implicit surfaces are visualized in real time thanks to a new sampling technique

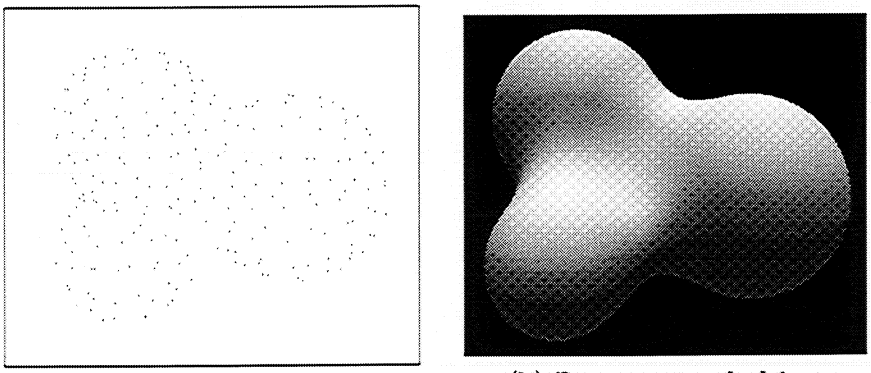
based on seeds-migration, that adapts at interactive rates to any surface deformation. Sample points are not directly used for display: the user can choose between displaying "scales" laying in each sample point tangent plane, or a piece-wise polygonization of the surface, where a polygon set is associated with each skeleton (this display mode has been used in Figure 8). The full system for interactive modeling and visualization of implicit surfaces is described in [TG94].

5 Experimental results

5.1 Validation of the algorithm on implicitly-defined objects

We have first validated the reconstruction algorithm on objets that had initially been created with implicit surfaces.

The object depicted in Figure 5 has been reconstructed with a single initial skeleton (thus validating the automatic subdivision process), and a single reconstruction window, including all the scene. Results show that the original skeletons and fields are correctly reconstructed by the process, without the creation of any extra skeleton.



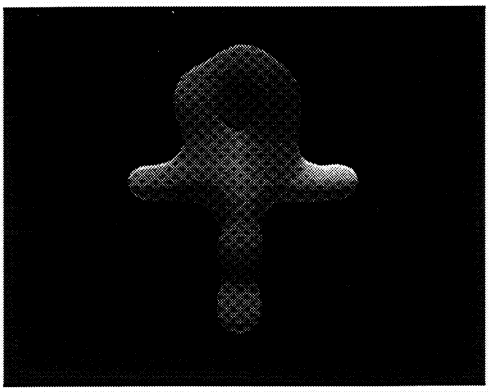
(a) Data points (b) Reconstructed object

Figure 5: Reconstruction of an iso-surface initially created with 3 punctual skeletons

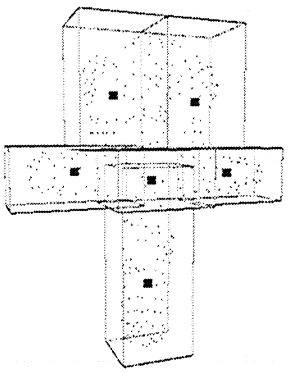
5.2 The use of reconstruction windows

The example depicted in Figure 6 illustrates the use of reconstruction windows on an object with hole and branches, also created with implicit surfaces.

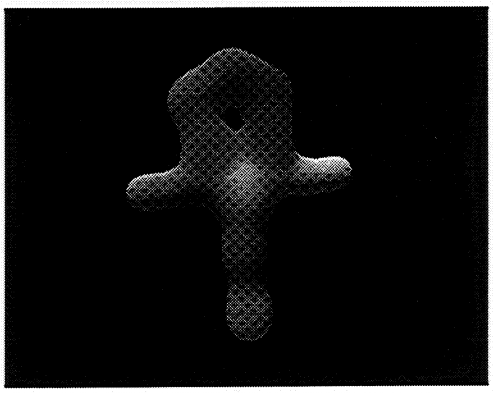
We have performed two different reconstructions based on the same initial skeleton: one with a single large window surrounding the whole scene (a), and the other with several reconstruction windows (c). Figure 6 (a) shows the first result: an approximation with 16 skeletons obtained after more than two hours of computations on an Indigo 2, R4400, 150 MHz workstation. Figure 6 (b), computed much more efficiently thanks to the local windows, is an approximation with 19 skeletons obtained in half an hour.



(a) Reconstruction with a single large window (about 2 hours)



(b) Interactive positioning of 6 local windows in the data space.



(c) Reconstruction with local reconstruction windows (about 30 mn).

Figure 6: The use of several windows for accelerating reconstruction

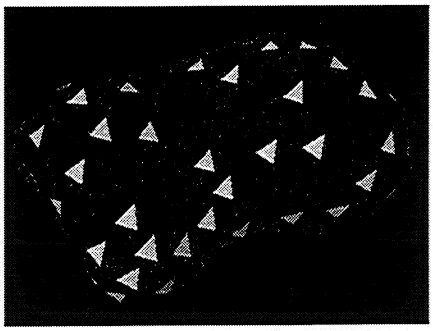
5.3 Reconstruction of medical organs

A kidney

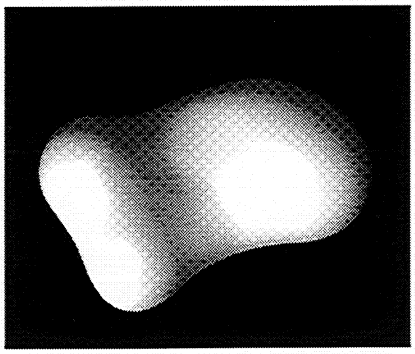
Figure 7(a) shows the data-points, arranged in a non-uniform repartition, and the reconstructed shape we get. 12 skeletons and 768 data points were used. The implicit surface is displayed using little surfacic elements laying on it.

A vertebra

A vertebra is a challenging example for validating the reconstruction process, due to it its very complicated geometry (a central hole and several small branchings). Figure 8(a) shows an intermediate step of the reconstruction with data points superimposed. Figure 8(b) shows a vertebra reconstructed from 2431 data points with 18 skeletons in 45 minutes. The compression ratio of the representation is about 1:100 (about 29000 bytes for the data set and only 360 for the implicit representation).

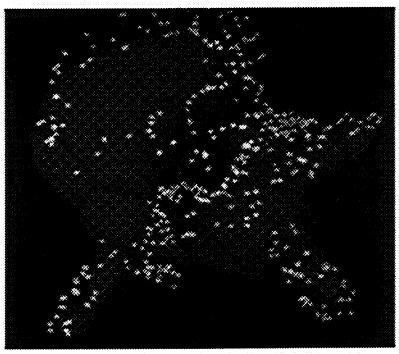


(a) Data points and resulting surface

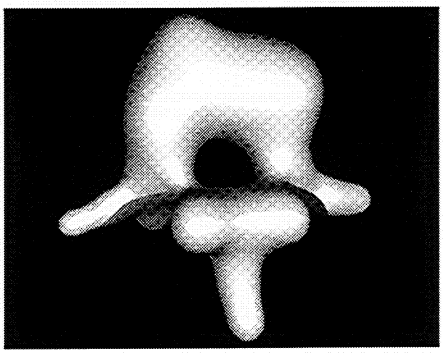


(b) Rendered version

Figure 7: Reconstruction of a kidney



(a) An intermediate step during the reconstruction of a vertebra



(b) Rendered version of a vertebra reconstructed with 18 skeletons

Figure 8: Reconstruction of a vertebra

6 Conclusion

We have presented a semi-automatic reconstruction method, that can be used on noisy scattered points of a medical organ. The method is based on implicit iso-surfaces generated by skeletons, that provides a smooth and compact representation of the surface. The basics is the minimization of an energy, that represents the distance between the data and the surface, during a series of skeleton refinements. We propose an efficient automatic way of selecting the skeleton to subdivide next, and a new method for optimizing reconstruction through the use of "reconstruction windows". The method is validated on both medical and non-medical data. The user can guide the reconstruction by initializing some skeletons and their reconstruction windows, thus taking benefits of his initial knowledge of the data.

An easy extension would be to enable the use of other kinds of skeletons generating field functions (such as segments) during the reconstruction process.

For some applications, an automatic generation of the initial skeletons may be more convenient that the interactive initialization process we have described.

We are currently studying an approach based on "medial axis" for the automatic positioning of initial skeletons and reconstruction windows.

Acknowledgements

We wish to thank Philippe Cinquin and Claude Puech who made this joint project possible. Many thanks to Stephane Lavallée and Jean-Dominique Gascuel for their technical support during this research.

References

[ABM94] D. Attali, P. Bertolino, and A. Montanvert. Using polyballs to approximate shape and skeletons. In 12^{th} *ICPR*, pages 626–628, October 1994.

[BG92] J.-D. Boissonnat and B. Geiger. Three-dimensional reconstruction of complex shapes based on the Delaunay triangulation. Report 1697, INRIA Sophia-Antipolis, Valbonne, France, 1992.

[Bli82] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, pages 235–256, July 1982.

[Boi84] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4), 1984.

[BS91] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

[BW90] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, March 1990.

[CC92] L.D. Cohen and I. Cohen. Deformable Models for 3D Medical Images using Finite Element and Balloons. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pages 592–598, June 1992.

[Del94] H. Delingette. *Modélisation et reconnaissance d'objets tridimensionnels 'a l'aide de maillages simplexes.* PhD thesis, Ecole Centrale Paris, France, July 1994.

[EM92] Herbert Edelsbrunner and Ernst P. Mucke. Three-dimensional alpha shapes. *1992 Workshop on Volume Visualization*, pages 75–82, 1992.

[Gas94] Jean-Dominique Gascuel. Fabule : un environnement de recherche pour l'animation et la simulation. In *Les Simulateurs, Troisième Séminaire du groupe de travail français Animation et Simulation*, October 1994.

[HDDH94] H. Hoppe, T. DeRose, T. Duchamp, and M. Halstead. Piecewise Smooth Surface Reconstruction. In *Computer Graphics (SIGGRAPH'94)*, pages 295–302, July 1994.

[HDDW92] H. Hoppe, T. DeRose, T.and McDonald J. Duchamp, and Stuetzle W. Surface reconstruction from unorganized points. In Catmull E. E., editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, pages 71–78, July 1992.

[KAW91] Zoran Kacic-Alesic and Brian Wyvill. Controlled blending of procedural implicit surfaces. In *Graphics Interface'91*, pages 236–245, Calgary, Canada, June 1991.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[LB94] J.O. Lachaud and E. Bainville. A discrete adaptative model following topological modifications of volumes. In *Discrete Geometry for Computer Imagery, Grenoble*, September 1994.

[Lei93] F. Leitner. *Segmentation dynamique d'images tridimensionnelles.* PhD thesis, Institut National Polytechnique de Grenoble, France, September 1993.

[MBE+91] J. V. Miller, D. E. Breen, Lorensen W. E., O'Bara R. M., and Wozny M. J. Geometrically deformed models: A method for extracting closed geometric models from volume data. In Sederberg T. W., editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, pages 217–226, July 1991.

[Mur91] Shigeru Muraki. Volumetric shape description of range data using blobby model. *Computer Graphics*, 25(4):227–235, July 1991.

[NHK+85] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Objects modeling by distribution function and a method of image generation (in japanese). *Trans. IEICE Japan*, J68-D(4):718–725, 1985.

[SS91] Pentland A. Sclaroff S. Generalized implicit functions for computer graphics. In Sederberg T. W., editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, pages 247–250, July 1991.

[TG94] Nicolas Tsingos and Marie-Paule Gascuel. Un modeleur interactif d'objets définis par des surfaces implicites. In *Secondes Journées de l'AFIG*, Toulouse, December 1994.

[TM91] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[TWK88] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.

[WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, pages 227–234, August 1986.

[WW89] Brian Wyvill and Geoff Wyvill. Field functions for implicit surfaces. *The Visual Computer*, 5:75–82, December 1989.

For modeling applications, the method enables to immediately visualize objects after any deformation, or any addition/suppression of a skeleton. This is essential for interactive design. Moreover, objects with specific unwanted blending properties can be designed as well, and stored with local bounding boxes for final rendering.

During interactive animation, the opaque piecewise display proves to be very convenient for observing collision and contact situations [Gas93, DG94]. Local boxes associated with each skeleton enable to accelerate collision detection, and they also permit collision processing between different parts of the same object, for which non-blending properties have been specified.

Work in progress includes attempts to improve the sampling repartition, which is good but not uniform. Sampling irregularities are not a real problem in modeling applications, but can lead to imprecise results during physically-based simulations, where seed points in contact areas are used for integrating response forces [Gas93].

Acknowledgements

Many thanks to Jean-Dominique Gascuel for the multiple discussions, for his help during the implementation, and for developing an efficient algorithm for direct ray-tracing on implicit surfaces.

References

[Bar81] Alan H. Barr. Superquadrics and angle-preserving transforms. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.

[Bli82] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, pages 235–256, July 1982.

[Blo88] Jules Bloomenthal. Polygonisation of implicit surfaces. *Computer Aided Geometric Design*, 5:341–355, 1988.

[BS91] Jules Bloomenthal and Ken Shoemake. Convolution surfaces. *Computer Graphics*, 25(4):251–256, July 1991. Proceedings of SIGGRAPH'91 (Las Vegas, Nevada, July 1991).

[BW90] Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116, March 1990.

[dFdMGTV92] Luiz Henrique de Figueiros, Jonas de Miranda Gomez, Demetri Terzopoulos, and Luiz Velho. Physically-based methods for polygonization of implicit surfaces. In *Graphics Interface'92*, pages 250–257, Vancouver, Canada, May 1992.

[DG94] Mathieu Desbrun and Marie-Paule Gascuel. Highly deformable material for animation and collision processing. In *5th Eurographics Workshop on Animation and Simulation*, Oslo, Norway, September 1994.

[Gas93] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics*, pages 313–320, August 1993. Proceedings of SIGGRAPH'93.

116

[Ton91] D. Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface'91*, pages 255–262, Calgary, Canada, June 1991.

[TPF89] Demetri Terzopoulos, John Platt, and Kurt Fleisher. Heating and melting deformable models (from goop to glop). In *Graphics Interface'89*, pages 219–226, London, Ontario, Canada, June 1989.

[WMW86] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, pages 227–234, August 1986.

Chapitre 5

Adaptive sampling of implicit surfaces

Ce chapitre détaille la méthode mise au point pour l'échantillonnage adaptatif efficace de surfaces implicites. L'approche proposée s'adapte aussi bien à la modélisation interactive, qu'à l'animation ou à la simulation.

La partie qui concerne les applications à la modélisation interactive a déjà fait l'objet d'une publication dans les actes d'un colloque français. L'article qui est joint, le second, a été accepté pour publication dans un atelier international :

[**TG94**] Modélisation interactive d'objets définis par des surfaces implicites. N. Tsingos et M.P. Gascuel. Dans les actes de *2-èmes Journées de l'AFIG*, Toulouse, décembre 1994.

[**DTG95**] Adaptive sampling of implicit surfaces for interactive modeling and animation. M. Desbrun, N. Tsingos and M.P. Gascuel. A paraître dans les actes de Implicit Surfaces'95. Grenoble, France, avril 1995.

Résumé

Ce papier présente une nouvelle méthode d'échantillonnage adaptatif pour les surfaces implicites. La méthode peut être utilisée aussi bien pour l'animation interactive que pour la modélisation. L'algorithme proposé échantillon les surfaces implicites engendrées par des squelettes et maintient efficacement ces points d'échantillonnage au cours du temps, même lorsque la surface change de topologie comme par exemple au cours de fractures ou de fusions. Deux modes complémentaires de visualisation temps réel découlent de la méthode: une visualisation basée sur des "écailles" tracées sur la surface, et une autre utilisant une polygonisation par morceaux. La méthode d'échantillonnage proposée est particulièrement bien adaptée à la prise en compte efficace des propriétés de "mélange indésirable" d'une surface (par exemple, les bras et les jambes d'un personnage modélisé par surfaces implicites ne doivent pas fusionner au cours du mouvement). De plus, elle peut être utilisée pour paver la surface à l'aide de boîtes englobantes locales, qui permettent d'optimiser la détection des collisions en animation, ainsi que le rendu final d'une surface par lancer de rayons.

Adaptive Sampling of Implicit Surfaces for Interactive Modeling and Animation

Mathieu Desbrun
Nicolas Tsingos
Marie-Paule Gascuel

iMAGIS / IMAG[1] - INRIA
BP 53, F-38041 Grenoble cedex 09, France

email: Mathieu.Desbrun@imag.fr

Abstract

This paper presents a new adaptive sampling method for implicit surfaces that can been used in both interactive modeling and animation. The algorithm samples implicit objects generated by skeletons and efficiently maintains this sampling, even when their topology changes over time such as during fractures and fusions. It provides two complementary modes of immediate visualization: displaying "scales" lying on the surface, or a piecewise polygonization. The sampling method is particularly well suited to efficiently avoid "unwanted blending" between different parts of an object (such as blending between a leg and an arm of an implicitly defined character). Moreover, it can be used for partitioning the implicit surface into local bounding boxes that will accelerate collision detection during animations and ray-intersections during high-quality final rendering.

1 Introduction

Implicit formulations of surfaces [Bar81, WMW86] have drawn a lot of attention during the past few years. Particularly convenient for modeling smooth objects of any topology and geometry such as objects with holes and branchings [WW89, BW90], they offer a compact storage of complex shapes and are well suited to direct ray-tracing algorithms [Bli82]. They appear as very promising for animating deformable solids, and have successfully been used in descriptive animation systems [WMG86, OM93], for animating metamorphosis [Wyv93], and for physically-based animation [PW89, TM91, Gas93, DG94, OM94] where their inside/outside function accelerates collision detection.

However, interactive modeling and animation with implicit surfaces has been limited by the difficulty of maintaining sample points on the surfaces at interactive rates. These points are necessary for visualizing implicit objects, and are also needed for detecting collisions and integrating response forces during physically-based animations [PW89, Gas93]. Insuring an adequate repartition of samples on the surface is important for both applications. But the problem is difficult since the surface topology may change over time: deformations, possible appearance of holes, fractures, and fusions at interactive rates requires

[1]IMAG is a Research institute affiliated with CNRS,Institut National Polytechnique de Grenoble, and Université Joseph Fourier.

being able to efficiently create and suppress sample points in the adequate areas. This paper presents a new solution to this specific problem, and presents some interesting fall-dawns of the method.

Background

Most of the existing techniques for sampling implicit surfaces are based on spatial partitioning polygonization. The first algorithm of this kind, known as *marching cubes*, was introduced in [WMW86], and later developed for medical imagery [LC87]. It consists in using an uniform grid to discretize space, find a cube that intersects the surface, and follow the surface by examining intersections with neighboring cubes. A set of polygons that approximate the surface is associated with each intersecting cube. More recent algorithms accelerate this method by the use of octrees, and take local curvature into account to reduce the number of polygons [Blo88, ZJ91, WG92, NB93]. However, these methods have two major drawbacks: they are compute-intensive so they cannot be performed in real time ; moreover, topological problems may raise during surface generation, leading to even more expensive processing. In addition, taking benefits of temporal coherence for accelerating the sampling of a surface that progressively moves and deforms would be a good idea, but it doesn't seem easy with these methods.

A completely different approach consists in scattering seed-points in space, and leaving them migrate to the implicit surface along the gradient direction [BW90]. Here, computations are much more efficient since no polygonization is computed. Moreover, this approach could take benefits of temporal coherence if the last set of sample points was used for initializing migration at the next time step. However, there is a severe drawback: the repartition of sample points on the implicit surface would become far from uniform after several deformations, and a topological change such as a fracture would require scattering points again, with the problem that there is no easy way to detect the fracture.

Providing an uniform repartition of sample points on an implicit surface can be done by connecting these points with interaction forces. A physically-based simulation of this "particle-system" leads to an equilibrium state where the repartition of samples is very good [dFdMGTV92]. A further improvement consists in introducing particles of adaptive size (where the size corresponds to the radius of the non-penetration area around a particle). After each deformation of the implicit surface, large particles are used for very quickly invading low sampled areas, reach equilibrium, and then subdivide until the desired precision is obtained. An application to interactive modeling is developed, where a few of these particles can be used for controlling the implicit surface's shape [WH94]. However, this adaptive sampling process remains computationally intensive, since a constrained optimization process is needed for enforcing the particles to stay on the implicit surface. Moreover, the number and degrees of freedom of the primitives that generate the iso-surface must be known in advance. Adding or suppressing one of them requires recomputing everything, which may be a problem for interactive design applications.

A last criteria for the choice of a sampling algorithm is the legibility it will

offer for implicit surface visualization. The first methods described compute polygonization of the surfaces, that are much more convenient for visualization that a mere set of points. Computing points without polygonization can be combined with the visualization of "scales", flat primitives centered on the sample points and oriented according to the surface tangent plane [WH94]. This gives a reasonably good idea on the shape of the object. However, this method doesn't provide any opaque visualization of implicit surfaces, which can be a problem for designing complex shapes and for displaying animations of colliding objects.

Overview

This paper presents a new and efficient algorithm for computing and maintaining sample points on an implicit surface, whose topology may change over time. The method takes benefits of temporal coherence, that is frequent in both animation and modeling fields (a designer drags primitives during a modeling process). Computed at interactive rates, the set of sample points enables an opaque display of the implicit surface thanks to piece-wise polygonization. Moreover, this method is well suited to the elimination of the unwanted blending problem (ie. modeling implicit objects with different parts that do not blend together), and enables to cover the surface with local bounding boxes that can be used for accelerating collision detection and ray-intersection algorithms.

Section 2 details the sampling algorithm. Sections 3 and 4 shows how one can take advantage of this approach for both interactive visualization and unwanted blending elimination. Section 5 describes further supplied optimizations. We conclude in Section 6.

2 An Efficient Adaptive Sampling Algorithm

Principle of the method

As pointed out in the introduction, performing a polygonal partitioning in real time seems impossible. A good solution for taking advantage of temporal coherence is to compute a set of points that migrate to the surface, as in the scattering method [BW90]. However, a good sampling repartition must be maintained. Our main ideas for attaining this goal are the following:

- Rather than creating and suppressing particles as in [WH94], we associate a fixed set of points, called "seeds" to each skeleton (or primitive) generating the iso-surface.
- Each seed moves along an axis that is fixed in the skeleton/primitive's local coordinate system. The use of a fixed axis rather than the gradient direction as in [BW90] has two benefits. Firstly, we don't need any intermediate gradient evaluation during a migration process. Secondly, a rigid motion of an implicit object during an animation will not require any seed re-computation, since they are stored in local referentials.

- During deformations and topological changes, a very simple validation/invalidation process insures a sufficiently good, although non-uniform, repartition of seed points on the surface.

The remainder of the paper describes the method in the terminology of implicit surfaces generated by skeletons [Bli82, NHK+85, WMW86, BW90, BS91]. The skeletons S_i can be any geometric primitive with a well defined distance function. S_i generates a scalar field f_i that decreases with the distance. The surface is an iso-surface for the sum of skeleton contributions:

$$Surface = P \ / \ f(P) = \sum f_i(P) = iso$$

where iso is a given isovalue. The surface normal is given by the field's gradient.

Initialization

The first step of the sampling algorithm consists in generating a set of axes that are well distributed around each skeleton: these axes will be used as seeds migration directions during the interactive modeling or animation process.

Any method could be used for initializing the axes. We are currently using the following process (see Figure 1):

1. We first extend the skeleton's bounding box by the thickness $e_i = f_i^{-1}(iso)$, that represents the thickness of the implicit volume when the skeleton is used alone.

2. We calculate a regular grid on the surface of this box that fits the desired sampling density, specified by the user.

3. We project each grid vertex on the skeleton to find the axis directions. A seed is initialized on each axis at the distance e from the skeleton.

This process does not provide an exact uniform repartition of axis directions around a skeleton, but is has the advantage of being quite simple and general. Anyway, a uniform axis repartition around a skeleton would not lead to a uniform sampling of the surface, due to the blending between several skeleton contributions.

Seeds migration

After each deformation due to interactive modeling or animation, seeds migrate along their axis to reach the iso-surface. This is done by finding an inner and an outer point along the axis (which can be done very efficiently due to temporal coherence), and then using binary search or interpolation for finding the new seed position.

Validation/Invalidation process

When several skeletons are blending together, seeds that penetrate into an area already sampled by another skeleton are not useful for the current surface

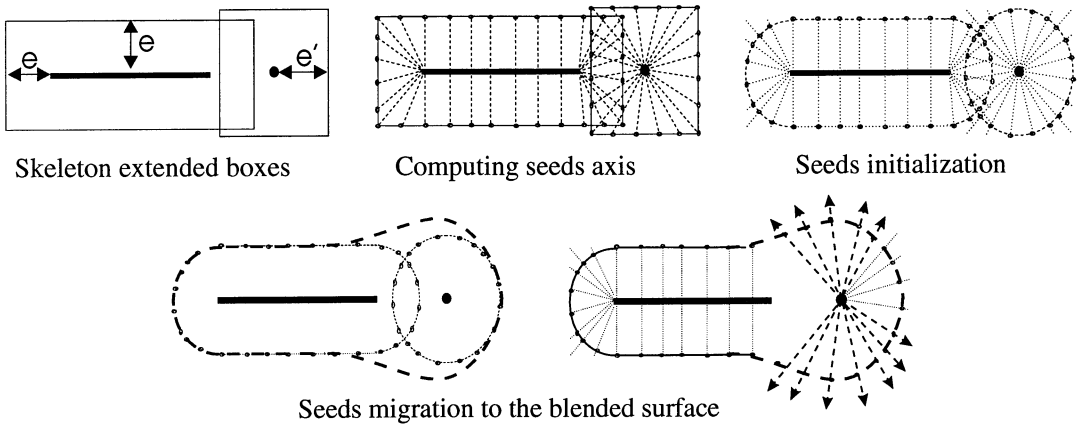


Figure 1: From initialization to migration: different stages of sampling

sampling (see Figure 1). However, these seeds should not be suppressed, since they may be needed later if the object happens to break into pieces. We choose to invalidate them for a while, according to the following criterion:

A seed located at P becomes invalid if the field f_i generated by its associated skeleton s_i is smaller than another field contribution. That is to say:

$$\{\exists j \neq i \mid f_j(P) > f_i(P)\}$$

Thanks to this criterion, only the necessary seeds points are computed on the surface at each time step, which saves computations, and their repartition is not too bad, although not uniform (see Figure 2). During deformations, invalid seeds may be validated again, so the sampling process efficiently adapts to geometrical and topological changes.

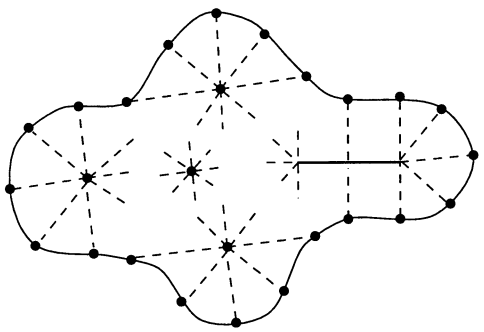


Figure 2: Valid seeds on an object defined by 5 skeletons.

Discussion

As stated earlier, the adaptive sampling method we have described does not result in a uniform distribution over the surface, contrary to [WH94] that used particle system simulations and constrained optimization techniques. However, our method samples implicit surfaces whose topology changes over time at

interactive rates. Moreover, skeletons can be added and suppressed during interactive modeling without perturbating our sampling process, contrary to the constrained optimization technique [WH94] where the number of skeleton had to be fixed in advance. The following section describes a method for opaque visualization that is a further benefit of our approach.

But it has to be noticed either that our initialization can only be used with simple primitives as points, segments or facets. When dealing with more complex skeletons, as spline-like or concave ones, care must be taken during the selection of axis direction. To ensure a reasonable distribution on every area, one can split the skeletons into simpler (segment-like and/or convex) ones. A further stage could be to simulate, always in the initialization phase, a particle system to obtain a uniform sampling. We have however thought that these calcutions were not essential in our case.

3 Interactive Visualization

Although well-suited to ray tracing, implicit surfaces cannot be rendered interactively at a high quality level. Cruder representations based on sample points need to be defined. We are currently using two complementary modes of interactive display that give good idea of the surface's shape.

Displaying scales

Visualizing a mere set of points is far from sufficient for giving a good idea of a surface. In [WH94], discs are placed at sample points with their normals aligned to the surface normals. Our system offers the same kind of representation: any graphic primitives - called scales for the analogy with fish scales - are placed on valid seed positions and oriented according to the surface normal. This gives a good idea of the object shape while demanding very few extra calculation and display time.

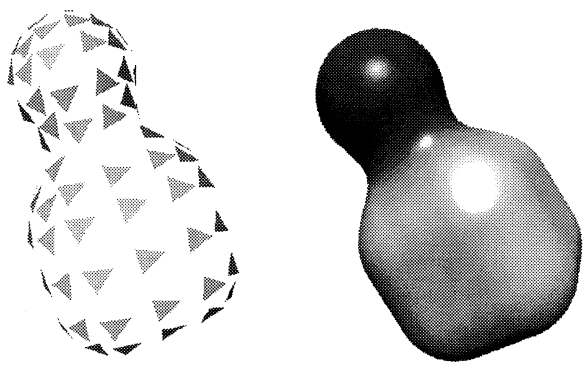


Figure 3: The same object with scales or ray-traced.

However, this visualization mode presents a major drawback in the case of complex scenes, and particularly for those, very frequent in animation, that represents objects in contact: dissociating objects that are in the foreground

and objects in the background is very difficult. In these situations, an opaque visualization would be more convenient.

Using a piecewise polygonization

We propose a second visualization mode, that is specific to our sampling technique, and offers an opaque vizualisation of the surface without any extra computation.

Rather that computing a Delaunay triangulation at each time step for connecting valid seeds from different skeletons, which would be compute intensive, we display a piecewise polygonization. During the sampling initialization process, a closed set of polygons constructed on seed points is associated with each skeleton. The topology of these meshes is given by the grids used for computing migration axis. During surface deformations, each mesh, based on both valid and invalid seeds, surrounds the implicit volume where a given skeleton is preponderant. The resulting opaque visualization is shown in figure 4. The fact that the polygonization used is piecewise creates noticeable discontinuities in the surface in the area where two skeletons blend together, but this is not a problem since the user can always switch to scales display to get a better idea of normal vectors in this particular area.

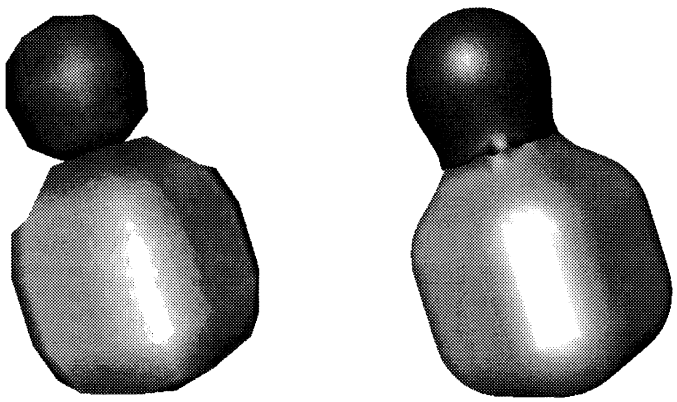


Figure 4: Two piecewise opaque visualizations with different sampling densities.

4 Avoiding Unwanted Blending

The unwanted blending case is a difficult problem known for a long time [WW89]. When we implicitly model characters for instance, we want their arms to blend with their shoulders, but not with another part of the body (Figure 5).

A solution, that was suggested in [WW89] and further developed in [OM93], consists in defining a neighboring graph between the different skeletons, and stating that a skeleton's field only blends with contributions from neighboring skeletons. More precisely, the field function f is replaced by the following pro-

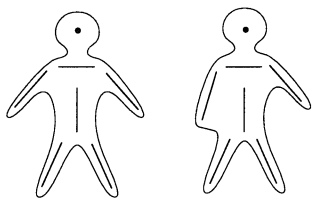


Figure 5: The unwanted blending problem

cedure for computing the field value at a point P:

1. compute all the field contributions at point P,
2. select the field f_i that is preponderant,
3. add the contributions from skeletons that are neighbors to S_i in the graph,
4. return the resulting value.

We must note that this algorithm can only been used with skeletons that have a finite radius of influence. Otherwise surface discontinuities would appear in areas where a skeleton that was used as a neighbor is not used anymore. Even with local field functions, blending graphs have to be defined with care.

Taking unwanted blending into account during interactive modeling and animation requires using this new procedural field function for recomputing sample points. However, computing all the field contributions at every seed point for only using the preponderant and neighboring fields would be compute intensive. Our specific adaptive sampling algorithm provides a much more efficient answer to this problem.

Sampling surfaces according to neighboring lists

We use the same initial idea as in [WW89, OM93] to cope with the problem: we define the blending graph by associating to each skeleton a list of neighbors to blend with. Consequently, the field value of a point in space is the sum of the predominant field and of the neighboring skeletons fields.

Fortunately, it happens that this type of calculation is made significantly easier when using our sampling algorithm. Indeed, each seed is sent by a particular skeleton and samples the area where the field of this skeleton is the highest. So we already know which is the preponderant field associated with any valid seed. This makes the computations very efficient: we just have to follow the list of neighboring skeletons and add their field contributions. Invalidation tests are made during the same process (the seed is invalidated if one of the neighboring contribution is higher that the first one).

Thanks to this algorithm, we are able to sample complex implicit objects, such as the articulated string depicted in Figure 6: the two extremities of this string do not blend together, but are able to collide during a physically-based

simulation.

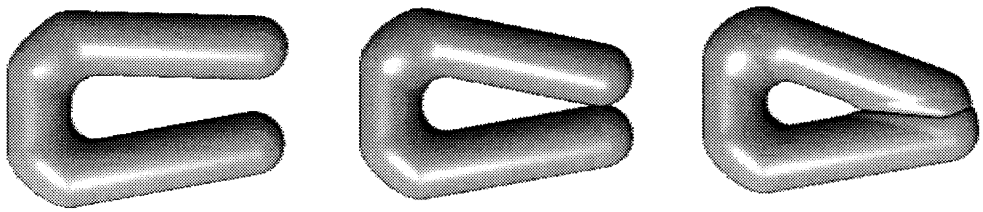


Figure 6: An articulated string whose extremities do not blend together

Benefits for further optimizations

In fact, storing lists of neighboring skeletons that define blending properties can be used for a more general purpose. Even if no unwanted blending property is needed, maintaining neighboring lists is a way to optimize computations.

After each deformation of an implicit surface, many seed points need to be recomputed. This can be done more efficiently if the list of skeletons that have a non-zero influence in the area is already known. We do this by computing current lists of neighboring skeletons: after each deformation, the skeletons whose influence areas overlap are said to be neighbors. Then current neighboring lists are used for every field computation at seed points, avoiding distance evaluations with distant skeletons.

5 Surface pavement with local bounding boxes

A last benefit of the adaptive sampling algorithm we have developed is to enable the definition of a hierarchy of bounding boxes that can be used for optimizing both collision processing and ray-intersection algorithms.

Local bounding boxes

The basics of the sampling algorithm was to imagine the implicit objects as an union of areas controlled by specific skeletons. This can be further exploited by associating an axis-parallel bounding box with the *valid seeds* of each skeleton. Computed from the seed positions, these boxes are enlarged by the maximal distance between sample points (derived from the sampling density defined by the user).

These local boxes provide a precise pavement of the implicit surface (*they don't necessarily cover the implicit volume*), and are used to compute a main bounding box that includes the entire surface. This two-level hierarchy of boxes, depicted in Figure 7, can be used for optimizing intersection tests.

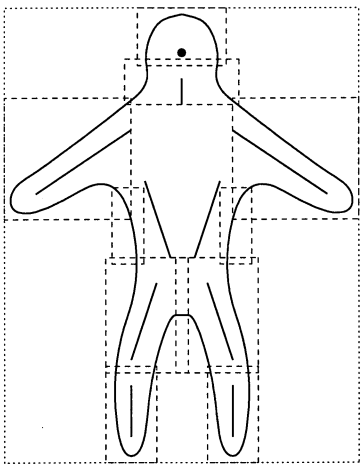


Figure 7: The hierarchy of bounding boxes associated with a surface

Application to collision detection

Local bounding boxes, that better fit the surface than the large one, can be used for optimizing collision detection during physically-based simulations. In particular, they unable to greatly reduce the number of field evaluations (see [DG94]).

The algorithm for detecting inter-penetration between two implicitly defined solids I_1 and I_2 is: if large bounding boxes B_1 and B_2 intersect, then, for each local box l_1 of I_1 that intersects B_2, is l_1 intersects at least one of I_2's local boxes, then test the seed points that have been used to define l_1 against I_2's field function. An inter-penetration is detected as soon as I_2's field on one of these seeds is greatest than the isovalue.

Application to final rendering

Final high-quality rendering on an implicit surface can be computed by processing direct ray-tracing, ray intersections with the surface being computed by binary search or linear interpolation.

In practice, we store the hierarchy of bounding boxes provided by our sampling algorithm together with the parameters that define the implicit surface (ie. positions of the skeletons, field parameters, and neighboring lists defining the blending properties). This accelerates both ray-intersection computations, and field evaluations, since a good candidate to be the "preponderant field" is known when a ray intersects a given local box. This leads to an optimized version of a ray-tracing that respects unwanted blending avoidance [?].

6 Conclusion

This paper has presented a new adaptive sampling algorithm for implicit surfaces. Objects of variable geometry and topology are sampled at interactive rates, the method taking benefits of temporal coherence. Applications include both modeling and animation/simulation fields.