Compression de données d'animation acquises par capture de mouvements

par

Philippe Beaudoin

Thèse de doctorat effectuée au

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Université de Montréal

Thèse présentée à la faculté des études supérieures de l'Université de Montréal

en vue de l'obtention du grade de Philosophiae Doctor (Ph.D.)

en informatique

Décembre 2007

Page d'identification du jury

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée

Compression de données d'animation acquises par capture de mouvements

présenté et soutenue à l'Université de Montréal par :

Philippe Beaudoin

a été évalué par un jury composé des personnes suivantes :

Président-rapporteur : et membre du jury	Neil F. Stewart, professeur au DIRO
Directeur de recherche :	Pierre Poulin, professeur au DIRO
Membre du jury :	Jian-Yun Nie, professeur au DIRO
Examinateur externe :	Marie-Paule Cani, professeur Université Joseph Fourier, Grenoble, France
Représentant du doyen : de la FÉS	François Prince, professeur, Dépt. de Kinésiologie

Résumé

La capture de mouvements permet de croquer sur le vif les gestes exécutés par un acteur et de les appliquer en 3D sur un personnage de synthèse. Cette technique rend possible l'acquisition rapide d'animations adaptées à une situation précise. Cet avantage peut toutefois s'avérer un inconvénient important en menant à une explosion de la quantité de données générées. Il devient donc nécessaire d'envisager différentes techniques permettant de réduire la taille de ces données.

Dans cette thèse, nous nous intéressons à la compression de banques de données acquises par capture de mouvements. Plus précisément, nous nous penchons sur les techniques de compression avec pertes exploitant les structures temporelles présentes dans ces données. Nous distinguons deux types de structures : les cohérences temporelles à petite échelle dues au fait que les signaux d'animation sont lisses, et les cohérences temporelles à grande échelle dues à la présence de mouvements fréquemment répétés dans la banque de données.

Dans un premier temps, nous montrons comment les techniques de compression à base de transformées en ondelettes peuvent être adaptées pour profiter des caractéristiques particulières des données d'animation. Ensuite, nous proposons un algorithme original combinant la segmentation et l'agrégation de mouvements et permettant d'identifier automatiquement des sous-séquences répétées à l'intérieur d'une banque de données. Nous introduisons aussi une technique, que nous nommons *PCA quantifiée* (principal component analysis), permettant d'exploiter les sous-séquences redondantes dans le cadre de la compression d'une banque de données.

Pour terminer, nous montrons comment l'identification des sous-séquences répétées rend possible la création de graphes compacts représentant le contenu d'une banque de données d'animation. Nous utilisons ensuite ces graphes pour synthétiser des mouvements originaux. Finalement, nous discutons de différentes façons de combiner ces structures et les techniques de compression introduites précédemment et montrons comment un tel amalgame pourrait éventuellement mener au développement de puissants outils de synthèse d'animations.

Mots-clés :

Infographie 3D, animation, capture de mouvements, compression, ondelettes, graphe de mouvements, groupes de mouvements, extraction de structure, synthèse d'animations

Abstract

Motion capture makes it possible to efficiently record the movements of an actor and to apply them to a synthesized 3D avatar. An advantage of this technique is its ability to quickly capture animations tailored for a specific purpose. However, this rapid process can lead to an explosion of the amount of generated data, making it essential to develop techniques aimed at reducing the size of this data.

In this thesis, we explore ways of compressing motion capture databases. More precisely, we look at lossy compression techniques that take advantage of the temporal structures visible in these databases. We distinguish between two types of structures: small scale temporal coherencies due to the smoothness of the animation signals, and large scale temporal coherencies due to the presence of repeated movements within the database.

We first show how wavelet-based compression techniques can be adapted to take advantage of the specificities of motion capture data. We then propose an original algorithm, based on a tightly coupled motion segmentation and clustering process, that makes it possible to automatically identify repeated subsequences within a database. We also introduce a technique, termed *quantized PCA* (principal component analysis), that takes advantage of these redundant subsequences for motion capture database compression.

Lastly, we show how these repeated subsequences can lead to compact graphs that illustrate the content of a motion database. We then use these graphs to synthesize original movements. Finally, we discuss various ways in which these structures can be combined with the previous compression techniques and show how such a combination could eventually lead to the development of powerful animation synthesis tools.

Key Words:

3D Computer graphics, animation, motion capture, compression, wavelet, motion graphs, motion bundles, structure extraction, animation synthesis

Table des matières

Liste des tableaux

Table des figures

À Jérôme et Claudiane, qui connaissent bien le mouvement.

À Ève-Marie.

Préface

Il est d'usage d'inclure, dans les premières pages d'une thèse, un court chapitre de remerciements. J'ai délibérément choisi de déroger à cette règle en rédigeant plutôt cette préface, qui me permet de délaisser un court instant le style d'écriture rigoureux, mais plutôt ennuyant, qui emplit la centaine de pages qui suivent. N'est-ce pas la meilleure façon de remercier la plupart des gens qui m'ont aidé et qui trouveront probablement plus de plaisir dans ce petit récit de mes aventures doctorales que dans les énoncés mathématiques qui hantent le reste de ce document ?

Je crois que la décision de passer du monde du travail au doctorat est rarement totalement rationnelle. Cette opinion peut surprendre, les scientifiques ayant la triste réputation de considérer l'intelligence et la raison comme les plus pures qualités, mais choisir de passer de la sécurité d'un emploi à l'inconnu et l'incertitude de la recherche requiert un brin de folie. De mon côté, cette décision est ancrée d'une part dans la vision romantique que je me faisais de l'enseignement universitaire. Bien que je souhaite toujours ardemment partager mon enthousiasme pour la science au sein d'une université, j'ai aujourd'hui une idée moins rose, et probablement plus exacte, de la tâche qui incombe à un professeur.

D'autre part, j'ai choisi le retour aux études dans l'espoir de trouver là un style de vie qui conviendrait mieux à mes ambitions familiales. En effet en novembre 2002, alors que j'alignais les heures supplémentaires pour tenter de terminer un jeu vidéo, je découvrais les joies de la paternité avec la naissance de notre fils Jérome. J'ai alors pensé que l'horaire flexible d'un étudiant me permettrait de passer beaucoup plus de temps avec lui. Si c'est en partie vrai, je n'avais alors pas envisagé qu'avec l'indépendance que je gagnais venait aussi une part importante de nouvelles angoisses. Aujourd'hui, alors que j'en suis à finaliser la rédaction de cette

thèse, les incertitudes atteignent un sommet. Comment mettre en valeur mes résultats ? Mes idées sont-elles présentées assez clairement ? Et après, qu'est-ce que je fais ?

À lire ce texte, on pourrait croire que les études doctorales offrent un avant-goût du purgatoire. N'y-a-t-il donc rien dans ce choix de vie ecentrique qui justifie de passer quatre ans entre les murs d'un sombre laboratoire ?

Il y a les gens. Quand on a la chance de rester assez longtemps dans un laboratoire universitaire on y voit forcément défiler une foule d'individus qui partagent tous l'originalité d'aimer l'école. Oubliez l'image du boutonneux à lunettes qui lève la main plus haut que les autres pour répondre au professeur. On est ici dans un monde éclectique et éclaté où règnent l'ouverture d'esprit et la pensée indépendante. De l'amateur de *Family Guy* au fan de musique japonaise en passant par la cinéphile, la nageuse élite et le collectionneur de bonsaï : à chaque détour se cache une conversation passionnante sur un sujet original.

Merci de m'avoir permis de perdre autant de temps à explorer des sujets souvent plus intéressants que l'infographie ! À Pierre-Marc Jodoin, mon voisin de bureau. Aux vieux routiers et futurs docteurs : Jean-François Duranleau, Di Jiang, Mélissa Jourdain, Luc Leblanc et Romain Pacanowski. Finalement, à tous ceux qui sont passés trop rapidement : Jean-François St-Amour, Nicolas Bergeron, Eric Bourque, Simon Bouvier-Zappa, Simon Clavet, Marie-Elise Cordeau, Martin Côté, Charles Donohue, Jean-François Dufort, Emric Epstein, Mathieu Gauthier, Martin Granger-Piché, Michelle Laprade, Mathieu Nesme, Fabrice Rousselle, Frédérik Rozon, Yann Rousseau, Yannick Simard et Dimitrios Touloumis.

Il y a ensuite les mentors. Ces professeurs dont la passion devient pratiquement un objet tangible et qui incarnent parfaitement cette idée romantique, mais ô combien inspirante, du chercheur constamment en quête de nouveaux savoirs. Je remercie tout d'abord Gilles Brassard, Pierre McKenzie et Alain Tapp. Bien qu'un certain sens pratique m'ait fait préférer l'infographie, je garde toujours un profond attachement pour l'informatique théorique, entretenu par les excellents cours et séminaires de ces chercheurs et présentateurs hors pairs.

Du côté du laboratoire d'infographie, je remercie Victor Ostromoukhov avec qui j'ai eu la chance de discuter aussi bien de ma future carrière que d'exotiques pavages du plan. Neil

Stewart, aussi, dont la rigueur mathématique m'a toujours inspiré et qui rédige présentement un manuscrit sur les surfaces de subdivision, que j'ai eu le plaisir et le privilège de lire.

Naturellement, je remercie mon directeur, Pierre Poulin, sans qui je n'aurais probablement jamais osé me lancer dans l'aventure du doctorat. Bien qu'il appelle constamment au dépassement de soi, Pierre reste un directeur très compréhensif sur qui on peut toujours compter. Ses contributions à mes apprentissages dépassent largement le contenu de cette thèse et son influence sur ma façon d'aborder la recherche universitaire est indéniable. Si j'ai un jour la chance de diriger des étudiants, il ne sera pas tout à fait étrangé à leurs souffrances... et à leurs succès !

Ce petit tour d'horizon ne serait pas complet si je n'y abordais pas le point tournant de mon doctorat : mon séjour à la University of British Columbia à l'été 2006. Ève-Marie, les enfants et moi avons en effet traversé le Canada en voiture pour nous installer pendant quatre mois dans une douillette résidence familiale sur le campus de UBC.

Là-bas, je me suis joint au laboratoire Imager où j'ai cotoyé bon nombre d'étudiants et de professeurs sympathiques. Pour leurs contributions à l'avancement de mes recherches, je remercie en particulier Roey Flor, Kevin Loken et Kang Kang Yin.

À UBC, j'ai aussi eu le privilège de travailler avec Michiel van de Panne, un expert de l'animation de personnages. Michiel est un chercheur hors pair, chez qui l'intuition n'a d'égale que la créativité. Les nombreuses discussions que nous avons eues ont fait faire un immense bond à mes travaux et ont mené à certaines des principales contributions présentées dans ce mémoire. À ses qualités de chercheur, Michiel combine des qualités humaines qui en font un excellent motivateur. J'espère avoir la chance de collaborer avec lui à nouveau dans le futur.

Si le récit qui précède offre un rapide aperçu de la vie d'un étudiant à l'intérieur d'une université, il omet cependant tout le reste. Quand la présentation approche, que le programme plante, que l'article est rejeté, que la recherche piétine, le microcosme universitaire n'est pas d'un grand support. C'est dans de telles situations que j'ai pu compter sur l'immense appui de ma famille et de mes amis. Je remercie plus particulièrement mes parents, des bouées immuables dans la tempête. Merci pour votre confiance et vos encouragements. Merci aussi à ma soeur

Catherine et son conjoint Sébastien, pour m'avoir fréquemment permis de m'évader hors de l'univers obnubilant de la recherche.

Merci infiniment à Ève-Marie, Jérôme et Claudiane. Pour les innombrables gestes de support, bien sûr, mais surtout pour le sens qu'ils donnent à ces quatre années. Si j'arrive aujourd'hui à destination, c'est grâce aux sourires quotidiens, aux mots d'encouragement, aux sorties en famille loin de mes tracas de chercheur. Quand la thèse sera écrite, que je l'aurai défendue et que tout sera terminé, ils seront toujours là. Et c'est tout ce qui m'importe vraiment.

Je remercie les différents organismes qui m'ont apporté une aide financière bien nécessaire. J'ai en effet pu profiter des bourses de doctorat ÉS B du CRSNG et B2 du FQRNT ainsi que d'une bourse de rédaction de l'Université de Montréal.

Cette thèse a été rédigée en grande partie dans la superbe bibliothèque municipale de Westmount. Un endroit unique et inspirant que je recommande à tous les auteurs qui, comme moi, ont tendance à manquer de discipline.

En terminant, une note sur les extraits en exergue de chapitres. Pour me donner du courage, j'entrecoupais mes épisodes de rédaction d'instants de lecture dans le parc voisin de la bibliothèque. Le dernier livre à m'avoir accompagné était *Jonathan Strange & Mr Norrell* de Susanna Clarke. Ce roman explore la quête de deux magiciens qui, dans une Angleterre fantastique des années 1800, tentent de retrouver les fondements d'une magie oubliée depuis longtemps. Pour ce faire, ils écument les bibliothèques, développent des théories, publient des articles et opposent leurs idées. Ce bouquin de 1000 pages, dont la lecture est presque aussi épique que l'écriture d'une thèse, m'est ainsi rapidement apparu comme le miroir fantastique de mes sobres aventures scientifiques. J'ai donc choisi d'illustrer mes chapitres par de brefs extraits de ce texte.

Philippe Beaudoin
Westmount, le 16 novembre 2007

Remarques

Droits d'auteur

Certaines images de ce document ont pu être tirées d'articles publiés dans des journaux ou des conférences scientifiques. La provenance des images est clairement indiquée dans les figures et la référence au travail est également fournie. Voici la note de droits d'auteur pour les travaux tirés d'une publication d'ACM :

> ACM COPYRIGHT NOTICE. Copyright © *YYYY* by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept, ACM Inc., fax +1 (212) 869-0481, or permissions@acm.org.

Voici la note de droits d'auteur pour les travaux tirés de *Graphics Interface* :

> Copyright © *YYYY* par l'Association canadienne de l'informatique. Il est permis de citer de courts extraits et de reproduire des données ou tableaux du présent compte rendu, à condition d'en identifier clairement la source.

Chapitre 1

Introduction

Lock a door against him and all that happens is that he learns first how to pick a lock and second how to build a better one against you !

Mr Norrell

L'évolution de l'informatique graphique s'est traduite de multiples façons, aussi bien en synthèse d'images qu'en animation. Aujourd'hui, par exemple, les animations de synthèse sont de plus en plus riches et détaillées. Ces animations se remarquent sur toutes sortes d'objets de natures très variées : un drapeau qui vole au vent, une fenêtre qui éclate en morceaux, la flamme d'une bougie qui vacille. Il reste néanmoins un type d'objets, omniprésent, dont l'animation représente un défi particulier : les personnages à forme humaine.

Bien qu'il existe différentes façons de synthétiser le mouvement d'un personnage, la capture de mouvements demeure sans doute la technique la plus populaire. Celle-ci consiste à habiller un acteur de marqueurs réfléchissants et à lui faire réaliser les mouvements désirés à l'intérieur d'une zone où la position des marqueurs sera régulièrement échantillonnée. Ces positions sont ensuite utilisées pour transposer le mouvement sur un squelette virtuel puis, finalement, sur l'*alter ego* synthétique de l'acteur.

L'utilisation de la capture de mouvements peut sembler contraignante en limitant les animations produites à celles pouvant être réalisées par un acteur dans un studio. La popularité de la technique s'explique cependant par le fait qu'elle permet de saisir de petits détails difficiles à synthétiser autrement. En fait, le succès de la capture de mouvements témoigne du fait que

le réalisme d'une animation d'un personnage nécessite souvent la présence de ces détails, dont l'absence serait immédiatement perçue par la plupart des observateurs.

Un autre avantage de la technique tient au fait que les technologies de capture de mouvements modernes permettent d'enregistrer rapidement une grande quantité de données. Il devient donc possible de saisir une scène entière dans laquelle un acteur enchaîne des mouvements très variés. Alternativement, on peut aussi préférer capturer de nombreuses variations sur un même mouvement.

En pratique, il n'est donc pas rare qu'on amasse une quantité impressionnante de données d'animation hétéroclites. Par exemple, la *CMU Motion Capture Database* [CMU] compte en ce moment plus de 1500 séquences d'animations humaines. Quant aux banques de données produites en entreprise, elles sont souvent encore plus imposantes. En fonction de l'usage qu'on souhaite en faire, ces données devront tantôt être stockées sur disque, tantôt transmises sur un canal à faible débit, ou même stockées dans la mémoire limitée d'une carte vidéo.

Ces différents scénarios montrent clairement que la réduction de la taille de stockage des animations est un problème qui mérite d'être étudié. Ce problème a cependant autant de facettes qu'il existe de façons différentes d'utiliser les animations de personnages. Avant de définir précisément l'objet de nos recherches, il nous apparaît donc intéressant de faire un rapide tour d'horizon des différentes formes que peut prendre la compression d'animations acquises par capture de mouvements.

1.1 Les différentes facettes du problème

La compression de n'importe quel type de données est un problème complexe et l'expérience a maintes fois démontré qu'il n'existait pas de solution unique s'adaptant à tous les scénarios. En compression d'images, par exemple, un photographe amateur préférera conserver un très grand nombre d'images de qualité plus faible alors qu'un professionnel préférera stocker moins d'images tout en conservant la qualité des photographies originales.

La compression de données acquises par capture de mouvements ne déroge pas à cette règle. Nous recensons donc différentes applications pratiques en nous penchant sur leurs besoins particuliers. Ce survol se veut délibérément général et vise à offrir une image d'ensemble de notre champ d'intérêt tout en constituant le préambule du problème étudié dans cette thèse et qui sera posé plus formellement par la suite.

1.1.1 Application à des banques de données

Une première gamme d'applications possibles de la compression d'animations de personnages est reliée à la croissance rapide de la taille des banques de données acquises par capture de mouvements.

Parmi ces applications on trouve l'archivage des banques de données dans le but d'en diminuer les coûts de stockage à long terme. Ici, on s'intéresse généralement à une compression n'engendrant aucune dégradation de façon à permettre la reconstitution ultérieure des données d'origine. Le développement d'une telle compression spécialisée pour les données de capture de mouvements ne semble cependant pas constituer un besoin criant. Ceci pour deux raisons principales : le très faible coût des médias de stockage à long terme ainsi que la relative efficacité des techniques génériques de compression sans pertes.

Une autre application possible concerne la réduction de la taille d'une banque de données dans le but d'en faciliter l'exploration et la manipulation. Dans ce contexte, la compression permettrait d'accélérer les différentes requêtes en réduisant la taille des données à accéder. Ici, le but visé n'est plus la réduction de l'espace mémoire total requis pour le stockage, mais plutôt une représentation des données à de multiples échelles permettant tantôt une vue d'ensemble plus grossière, tantôt une vue précise d'une animation particulière. Ainsi, dans ce contexte, les échelles plus larges pourront être compressées moins fidèlement tandis que l'échelle la plus précise devra être compressée sans pertes.

Finalement, un dernier problème associé aux banques de données consiste à transmettre les informations qu'elles contiennent à travers un canal à débit limité, comme l'internet. Ici, les problèmes qui se posent sont similaires à ceux rencontrés pour l'exploration des données. Notons cependant un défi supplémentaire : l'intérêt que représente la transmission en continu (*streaming*). En effet, la transmission en continu permet d'afficher les premiers instants d'une animation avant même qu'elle ne soit entièrement reçue.

Si, à long terme, les défis précédents pourront légitimement se poser, force est de constater que les besoins actuels en terme de compression de banques de données sont somme toute limités, voire inexistants. Ils n'en demeurent pas moins des champs d'application potentiels des techniques étudiées dans cette thèse et, en tant que tels, méritent d'être conservés en mémoire.

1.1.2 Utilisation dans le cadre d'applications interactives

L'archivage à long terme de données constitue l'extrémité d'une importante hiérarchie de formes de stockage. À l'opposé on retrouve des mémoires à accès rapide dédiées au stockage à très court terme. Parmi celles-ci, citons par exemple les différents niveaux de mémoire cache de l'unité centrale (UCT) ou encore la mémoire dédiée disponible sur une carte d'accélération vidéo. La mémoire vive principale, quant à elle, se trouve à mi-chemin entre ces deux pôles.

Le stockage des données d'animation sur ces différentes formes de mémoire pose aussi un certain nombre de problèmes, en particulier dans le cas d'applications interactives qui souhaitent maximiser l'utilisation des ressources disponibles tout en présentant simultanément un grand nombre d'animations. L'exemple des jeux vidéo est probablement le plus éloquent. En fait, l'absence d'une compression adéquate et le grand nombre de personnages animés font en sorte que les animations constituent parfois la majorité des données stockées [Lav04].

Un autre facteur explique aussi le besoin grandissant de techniques de compression d'animations dans ces applications. Celui-ci tient au déséquilibre entre l'augmentation très rapide de la puissance de calcul des unités de traitement et l'augmentation plus modeste de l'espace mémoire disponible [Wal07]. Ainsi, sans compression, l'efficacité d'un système d'animation pourra être limitée par la difficulté de contenir les données dans la mémoire cache alors même que l'unité de traitement n'est pas exploitée à sa pleine capacité. Ce problème se présente de façon encore plus importante sur les plateformes portables comme les organisateurs personnels, les téléphones cellulaires ou les consoles de jeu portatives.

Ce type de déséquilibre est un exemple typique d'environnement dans lequel la compression s'avère très efficace. Celle-ci a en effet comme conséquence de transférer la charge de la mémoire vers l'unité de traitement. À cet effet, plusieurs entreprises de développement de jeux vidéo ont d'ailleurs mis au point des techniques de compression, généralement basées sur un simple sous-échantillonnage ou sur la quantisation des données [Dun04]. Dans le contexte de ce type d'applications, l'objectif de la compression n'est plus de conserver l'intégrité des données mais plutôt d'en réduire au maximum la taille tout en permettant de reproduire une animation perceptuellement fidèle à l'originale. De plus, le temps de calcul requis pour la décompression doit être faible de façon à ne pas renverser le problème en surchargeant l'unité de traitement.

Ainsi, si la compression apparaît de prime abord comme une façon de réduire la taille des grandes banques de données, elle peut aussi jouer un rôle crucial en permettant de maximiser l'utilisation de la mémoire cache dans les applications interactives.

1.1.3 Utilisations connexes

Pour terminer ce tour d'horizon, nous présentons certaines utilisations connexes qui, bien que ne profitant pas directement de la taille réduite des données, peuvent bénéficier des informations acquises lors de la compression.

Par exemple, la compression de données comporte parfois une étape de détection des redondances. Dans le cas de la capture de mouvements, cette étape vise à identifier les sous-séquences similaires dans une banque de données. La simple identification de ces sous-séquences impose une structure de graphe à l'ensemble des données en permettant d'identifier les groupes de sous-séquences similaires qui peuvent être juxtaposés.

De tels graphes ont une importance particulière en synthèse de mouvements. Ils sont utilisés, par exemple, dans plusieurs applications interactives de façon à diriger un personnage animé. Bien qu'il existe déjà des techniques pour extraire ces graphes [KGP02], aucune n'est basée sur l'identification automatique de sous-séquences similaires.

De la même façon, on peut imaginer utiliser un ensemble de sous-séquences similaires de façon à décrire une famille complète de mouvements. Par exemple, une banque de données peut contenir un grand nombre d'animations de courses généralement similaires mais dont les styles ou les détails varient. Après avoir identifié une famille de mouvements, il devient possible d'extraire un certain nombre de paramètres permettant de la décrire et, ainsi, de synthétiser un nouveau mouvement appartenant à cette famille. De telles techniques de paramétrisation de mouvements ont déjà été mises au point [SO06, HG07] mais, encore une fois, aucune ne s'appuie sur le type d'identification de similarités qui pourrait être utilisée dans le cadre de la compression.

Ainsi, bien qu'ils n'en constituent pas le but premier, nos travaux en compression pourront présenter des particularités originales susceptibles d'être appliquées en synthèse de mouvements. Il pourra ainsi être intéressant d'évaluer le potentiel de ces techniques dans le cadre des applications connexes que nous venons d'énumérer.

Ce rapide survol, bien que général, nous permet de constater l'étendue des applications possibles de la compression d'animations. Ainsi, en vue de bien définir le problème à l'étude, nous pourrions choisir de nous attaquer à une de ces applications. Cependant, le peu de résultats en compression d'animations nous fait préférer une approche plus générale. Nous favoriserons donc une étude à haut niveau du problème de la compression menant au développement d'une

banque d'outils et de connaissances qui pourront ultérieurement être utilisés dans le cadre des différentes applications précédemment énumérées.

1.2 Définition du problème étudié

Le domaine de la compression de données acquises par capture de mouvements est vaste et le peu de recherche qui lui a été consacrée fait qu'il est mal jalonné. Il apparaît donc préférable de défricher ce territoire avant de s'attaquer à une application particulière. Dans cette optique, l'approche adoptée dans cette thèse consiste à étudier certains aspects théoriques reliés à la compression de données acquises par capture de mouvements, le tout complémenté par des implantations prototypes permettant l'analyse des résultats.

Dans cette section, nous nous attardons ainsi à définir le problème à l'étude en identifiant les différentes approches possibles quant à la compression et en soulignant celles qui ont été retenues dans le cadre de cette thèse. Contrairement à la section précédente, nous nous attardons ici aux aspects théoriques qui régissent la compression et non pas aux applications qu'on peut en faire.

1.2.1 Compression avec ou sans pertes

On distingue en général deux types de compression : la compression sans pertes qui permet de récupérer l'intégralité des données originales, et la compression avec pertes qui, au contraire, perturbe les données de manière irréversible. L'intérêt de la compression avec pertes vient du fait que, pour certains types de données, il peut être possible d'introduire des perturbations qui ont une influence minime sur la qualité du résultat tout en permettant d'augmenter substantiellement le taux de compression et/ou les performances des algorithmes de compression et de décompression.

Dans le cas des données acquises par capture de mouvements, ces deux types de compression sont possibles. En particulier, la compression avec pertes peut être envisagée étant donné que les animations sont généralement destinées à être visualisées par un individu. En effet, il est bien connu que la perception humaine est limitée et que l'introduction de certaines formes de perturbations pourrait ne pas affecter la qualitée perçue.

Dans le cadre de cette thèse nous aborderons uniquement les formes de compression avec pertes. Ce choix se justifie d'une part par l'intérêt plus important pour les applications utilisant

ce type de compression et, d'autre part, par le fait qu'une compression avec pertes permet de mieux exploiter la nature des données.

1.2.2 Types de corrélations

Les données d'animation acquises par capture de mouvements présentent deux types de correlations susceptibles d'être exploitées dans le cadre d'une compression. Tout d'abord, les données présentent une cohérence temporelle. C'est-à-dire que l'état d'un personnage animé varie de manière graduelle dans le temps. D'autre part, les différents degrés de liberté du personnage sont corrélés. Il est possible, par exemple, que le mouvement du coude ait tendance à suivre celui de l'épaule.

Bien que ces deux types de corrélations présentent un intérêt certain dans le cadre d'une compression avec pertes, ils n'en représentent pas moins des axes de recherche distincts. Dans le but de mieux circonscrire nos travaux, nous avons ainsi préféré mettre l'emphase sur l'étude des cohérences temporelles qui feront l'objet de la majeure partie de cette thèse.

Les corrélations entre les degrés de liberté seront néanmoins introduites sommairement de façon à constituer une base de comparaison pour les résultats présentés ainsi que pour démontrer l'intérêt de certaines techniques exploitant les cohérences temporelles.

1.2.3 Échelle des cohérences temporelles

Les cohérences temporelles se présentent à plusieurs échelles. La cohérence à très petite échelle vient du fait que la position du personnage à un temps donné soit très similaire à sa position un court instant plus tôt. À plus grande échelle, on remarquera, par exemple, que l'animation de marche est un cycle où chaque répétition est très similaire à la précédente. À une échelle encore plus importante, on retrouvera parfois dans une séquence plusieurs répétitions, à des temps différents, d'un même mouvement, voire d'une même série de mouvements.

L'exploitation de ces différentes échelles de cohérences temporelles pose un certain nombre de défis distincts dont l'étude constitue l'essentiel des travaux de recherche présentés dans cette thèse. Nous nous pencherons donc aussi bien sur la compression de courtes animations présentant des cohérences à petite échelle que sur la détection et l'exploitation des redondances à grande échelle dans une banque de données plus importante.

En résumé, le problème étudié dans cette thèse peut se décrire comme l'étude des cohérences temporelles à plusieurs échelles dans le cadre de la compression avec pertes de données d'animation acquises par capture de mouvements.

Nous débutons cette étude au chapitre 2 par une brève exposition du vocabulaire et des notations mathématiques qui seront utilisées dans le cadre de cette thèse. Au chapitre 3, nous proposons une revue des travaux antérieurs en compression d'animations acquises par capture de mouvements. Vu le peu de publications en rapport direct avec notre sujet de recherche, nous nous attardons aussi sur les résultats dans le domaine de la compression de maillages animés ainsi que sur les travaux reliés à l'animation par capture de mouvements susceptibles d'être exploités dans nos recherches.

Les quatre chapitres suivants constituent le coeur de nos recherches. Au chapitre 4 nous introduisons une technique visant à exploiter les cohérences à petite échelle. Cette technique, basée sur les ondelettes, est spécifiquement adaptée aux données acquises par capture de mouvements. Le chapitre 5 introduit une technique automatique d'extraction de sous-séquences redondantes présentes dans une banque de données acquises par capture de mouvements. Le chapitre 6, quant à lui, intègre les notions précédentes dans le cadre d'une technique de compression exploitant les redondances à grande échelle. Finalement, le chapitre 7 montre comment l'identification des sous-séquences redondantes peut être utilisée pour synthétiser des animations originales et suggère une utilisation de la compression dans ce contexte.

Nous concluons cette thèse au chapitre 8 par un rappel de nos contributions ainsi que par une exposition des directions possibles pour les recherches futures dans ce domaine.

Chapitre 2

Préliminaires

How can we restore English magic until we understand what it is we are supposed to be restoring ?

Jonathan Strange

Les concepts que nous abordons dans la suite de cette thèse s'inscrivent tous dans le domaine général de l'analyse de données acquises par capture de mouvements. Bien que les notions récurrentes de ce domaine apparaissent dans de nombreuses publications, il est difficile d'en extraire un vocabulaire ou un ensemble de notations mathématiques consensuels. Dans cette thèse, la tâche est d'autant plus difficile que les publications en français traitant de ce sujet sont très rares.

Nous avons donc pris le parti d'établir une convention et un vocabulaire unifiés et de les utiliser dans l'ensemble de ce document. Ce chapitre ne se veut pas une introduction aux concepts fondamentaux de l'animation par capture de mouvements, mais plutôt un outil de référence pour clarifier l'usage spécifique que nous faisons de certains termes. Nous supposons ainsi que le lecteur est familier avec le domaine.

2.1 Notation mathématiques

La notation mathématique adoptée dans cette thèse vise à s'ajuster aux conventions d'usage tout en favorisant la lisibilité. Nous utilisons ainsi la convention générale du tableau 2.1 pour l'attribution des noms de variables. Certaines variables sont utilisées tout au long du document pour représenter des éléments spécifiques (tableau 2.2). Nous utilisons aussi les opérations données au tableau 2.3.

$\mathbb{R}$	Ensemble des réels
$\mathbb{N}$	Ensemble des naturels
$a, b, c, \alpha, \beta, \gamma, ...$	Scalaire ou signal scalaire
i, j, k, l, m, n, o	Scalaire ou signal scalaire $\in \mathbb{N}$
$r, s, t, u, v, w, x, y, z$	Scalaire ou signal scalaire $\in \mathbb{R}$
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \boldsymbol{\mu}, ...$	Vecteur ou signal vectoriel (vecteurs colonne)
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \boldsymbol{\Sigma}, ...$	Matrice
$\mathbf{I}$	Matrice identité
$A, B, C, ...$	Ensemble ou espace vectoriel
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \Pi, ...$	Relation ou distribution de probabilité
$a(t), b(t), \mathbf{a}(t), \mathbf{b}(t), ...$	Signal scalaire ou vectoriel évalués au temps t
$a(i), b(i), \mathbf{a}(i), \mathbf{b}(i), ...$	Signal scalaire ou vectoriel, échantillon d'indice i
$a[i, j], \mathbf{a}[i, j]$	Sous-signal composé des échantillons $i, i + 1, ..., j$

TAB. 2.1 – Conventions mathématiques générales.

Un des problèmes que nous avons éprouvés lors de la rédaction de cette thèse consistait à déterminer une notation non ambiguë permettant à la fois de rapprocher et de distinguer les différents concepts que nous abordons. Par exemple, le nombre de degrés de liberté est un concept qui se rapproche beaucoup du nombre de joints dans un squelette, et il semblait inapproprié de réserver deux variables pour les distinguer. Nous avons ainsi choisi d'apposer des étiquettes aux différentes variables, tel que le montre le tableau 2.4. L'utilisation systématique de ces étiquettes introduirait cependant une lourdeur indésirable dans les équations, d'autant plus qu'elles s'avèrent souvent superflues dans le cadre limité d'un chapitre. Nous les omettons donc lorsque le contexte est assez clair, tout en prenant soin de mentionner ces omissions dans une note de bas de page en début de chapitre.

2.2 Terminologie particulière

Cette section s'attarde à introduire la terminologie et à souligner l'usage particulier que nous en faisons. Nous avons préféré cette approche à l'utilisation d'un glossaire car elle nous permet d'organiser les concepts logiquement en illustrant les liens qui les unissent. Pour permettre de les retrouver facilement, les termes définis dans la suite de ce chapitre sont indiqués en caractères gras.

$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Base euclidienne 3D
k	Nombre de dimensions dans un vecteur
n, m	Nombre d'échantillons dans un signal ou de caractères dans une chaîne
t	Temps
r	Taux de compression
i, j, o	Indices
s	Signal grossier lors d'une transformation en ondelettes
d	Signal de détail lors d'une transformation en ondelettes
$\psi_i(t)$	i-ième fonction d'une base d'ondelettes
$\mathbf{d}$	Vecteur correspondant à une pose
$\mathbf{d}(t), \mathbf{d}(i)$	Signal correspondant à un mouvement
$\mathbf{d}[i, j]$	Signal correspondant à une sous-séquence
$\hat{\mathbf{d}}$	Pose approximative reconstruite après compression
$\mathbf{w}$	Vecteur de pondérations pour chaque degré de liberté
c	Caractère
$c(i)$	Chaîne de caractères
$c[i, j]$	Sous-chaîne de caractères
b	Valeur binaire
$b(i)$	Signal binaire
E	Espace ou sous-espace vectoriel
$\mathbf{e}(i)$	i-ième vecteur de base de E
$\Pi, \mathcal{D}$	Distribution de probabilité
$\mathcal{G}$	Distribution gaussienne
σ	Variance
$\boldsymbol{\mu}$	Moyenne vectorielle d'une distribution ou d'un ensemble
$\boldsymbol{\Sigma}$	Matrice de covariance d'une distribution
C	Groupe de sous-chaînes (ensemble de sous-chaînes)
M	Groupe de mouvements (ensemble de sous-séquences)

TAB. 2.2 – Conventions mathématiques spécifiques.

$\mathbf{a}_i$	i-ième composante du vecteur $\mathbf{a}$, ou i-ième signal scalaire du signal $\mathbf{a}$
$\mathbf{A}_{i,j}$	Valeur de la cellule (i, j) de la matrice $\mathbf{A}$
$\mathbf{a}^\mathsf{T}, \mathbf{A}^\mathsf{T}$	Transposée de $\mathbf{a}$ ou $\mathbf{A}$
$\lvert\mathbf{a}\rvert$	Norme euclidienne du vecteur $\mathbf{a}$
$\langle\mathbf{a}, \mathbf{b}\rangle$	Produit scalaire de $\mathbf{a}$ et $\mathbf{b}$
$\mathrm{span}(A)$	Sous-espace vectoriel sous-tendu par les vecteurs de l'ensemble A
$\mathrm{proj}(E, \mathbf{a})$	Projection du vecteur $\mathbf{a}$ sur le sous-espace vectoriel E
$E^\perp$	Complément orthogonal du sous-espace E
$A(i)$	i-ième élément de l'ensemble A
$\lvert A\rvert$	Cardinalité de l'ensemble A
$\bar{A}$	Moyenne des éléments de l'ensemble A
$\mathrm{quant}(A, s)$	Quantile s des éléments de l'ensemble A
$\tilde{C}, \tilde{M}$	Largeur d'un groupe de sous-chaînes ou de mouvements (sections 5.5 et 5.6)
$\mathrm{vol}(C), \mathrm{vol}(M)$	Volume d'un groupe de sous-chaînes ou de mouvements (sections 5.5.3 et 5.6)
$\dot{\mathbf{a}}$	Signal $\mathbf{a}$ dérivé par rapport au temps
$\mathrm{trunc}(a, i)$	Signal a reconstruit en conservant i échantillons d'ondelettes (section 4.2)
$\mathrm{split}(a)$	Séparation du signal a en ses échantillons pairs et impairs
$\mathrm{merge}(a, b)$	Jonction des signaux a et b en intercalant les échantillons
$\mathcal{P}(a)$	Opérateur de prédiction appliqué au signal a
$\mathcal{U}(a)$	Opérateur de mise à jour appliqué au signal a
$\mathcal{O}(f)$	Ensemble des fonctions dans l'ordre de la fonction f
$\mathcal{D}(\mathcal{G}_1, \mathbf{v})$	Distance de Mahalanobis entre la gaussienne $\mathcal{G}_1$ et point $\mathbf{v}$ (section 5.4.6)
$\mathcal{D}(\mathcal{G}_1, \mathcal{G}_2)$	Distance de Mahalanobis entre les gaussiennes $\mathcal{G}_1$ et $\mathcal{G}_2$

TAB. 2.3 – Opérateurs mathématiques utilisés.

$n^{\mathtt{D}}$	Nombre d'échantillons dans un signal correspondant à une animation
$n^{\mathtt{S}}$	Nombre d'échantillons dans un signal correspondant à une sous-séquence
$n^{\mathtt{C}}$	Nombre de coefficients d'ondelettes dans une sous-séquence compactée (section 6.2.3)
$n^{\mathtt{M}}$	Nombre d'éléments dans un ensemble de vecteurs de sous-séquences (section 6.2.3)
$m^{\mathtt{D}}$	Nombre de caractères dans une chaîne
$m^{\mathtt{S}}$	Nombre de caractères dans une sous-chaîne
$\mathbf{d}^{\mathtt{D}}$	Vecteur de poses ou signal contenant directement les degrés de liberté
$\mathbf{l}^{\mathtt{D}}$	Longueur des membres associés à chaque degré de liberté de $\mathbf{d}^{\mathtt{D}}$
$k^{\mathtt{D}}$	Nombre de degrés de liberté, nombre de dimensions du vecteur $\mathbf{d}^{\mathtt{D}}$
$\mathbf{d}^{\mathtt{P}}$	Vecteur de poses ou signal contenant la position des différents joints
$\mathbf{l}^{\mathtt{P}}$	Longueur des membres associés à chaque élément de $\mathbf{d}^{\mathtt{P}}$
$k^{\mathtt{P}}$	Nombre de dimensions du vecteur $\mathbf{d}^{\mathtt{P}}$
$\mathbf{d}^{\mathtt{I}}$	Vecteur de poses ou signal indépendant de la position du personnage (section 5.4.1)
$k^{\mathtt{I}}$	Nombre de dimensions dans $\mathbf{d}^{\mathtt{I}}$
$\mathbf{d}^{\mathtt{M}}$	Vecteur représentant une sous-séquence (section 6.2.3)
$k^{\mathtt{M}}$	Nombre de dimensions dans $\mathbf{d}^{\mathtt{M}}$
$\mathbf{m}^{\mathtt{D}}$	Nombre de coefficients d'ondelettes à conserver avant optimisation (section 4.4.2)
$\mathbf{m}^{\mathtt{P}}$	Nombre de coefficients d'ondelettes à conserver après optimisation (section 4.5.1)
$\varepsilon^{\mathtt{A}}$	Mesure de distorsion angulaire
$\varepsilon^{\mathtt{D}}$	Mesure de distorsion angulaire pondérée
$\varepsilon^{\mathtt{P}}$	Mesure de distorsion positionnelle
$\varepsilon^{\mathtt{M}}$	Mesure d'erreur dans la compression de groupes de mouvements (section 6.2.4)
$k^{\mathtt{PCA}}$	Nombre de dimensions conservées après une projection *PCA* (section 4.8.4)
$r^{\mathtt{PCA}}$	Taux de compression lors d'une compression par *PCA* (section 4.8.4)
$r^{\mathtt{Q}}$	Taux de compression lors d'une compression par quantification (section 4.8.2)
$r^{\mathtt{SE}}$	Taux de compression lors d'une compression par sous-échantillonnage (section 4.8.3)
$r^{\mathtt{W}}$	Taux de compression lors d'une compression par ondelettes (section 4.4.2)
$r^{\mathtt{IK}}$	Taux de compression lors d'une compression par cinématique inverse (section 4.6)

TAB. 2.4 – Étiquettes de variables utilisées.

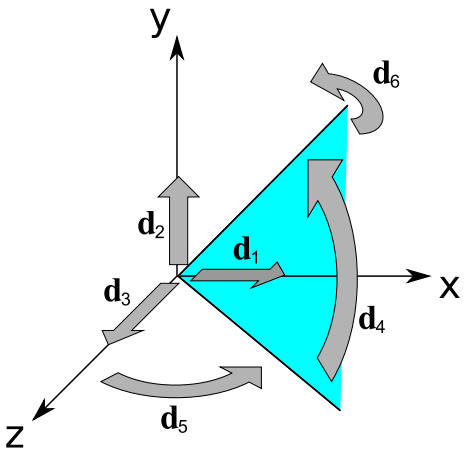


FIG. 2.1 – Les différents degrés de liberté attachés à la racine : les translations $\mathbf{d}_1^{\mathtt{D}}, \mathbf{d}_2^{\mathtt{D}}, \mathbf{d}_3^{\mathtt{D}}$ et les angles de rotation $\mathbf{d}_4^{\mathtt{D}}, \mathbf{d}_5^{\mathtt{D}}, \mathbf{d}_6^{\mathtt{D}}$.

2.2.1 Configuration d'un squelette

Un **squelette** est un ensemble hiérarchique de **joints** organisés en une arborescence débutant au **joint racine**. Au joint racine est associé un référentiel décrit par une transformation affine rigide (rotations et translations) permettant de positionner le squelette dans l'espace. À chacun des autres joints correspond un **os**, soit une transformation affine rigide décrivant précisément la position du joint à l'intérieur de son parent ainsi que son orientation au repos. On définit aussi la **longueur d'un os** comme la distance entre un joint et son parent.

Au joint racine correspondent six **degrés de liberté** : la position 3D et trois rotations autour des axes cartésiens, illustrés à la figure 2.1. À chacun des autres joints correspondent de 0 à 3 degrés de liberté permettant des rotations autour des axes cartésiens. Une **pose** du squelette peut être obtenue en spécifiant une valeur pour chacun des degrés de liberté. Une pose ainsi décrite se représente sous la forme d'un vecteur

$$\mathbf{d}^{\mathtt{D}} = (\mathbf{d}_1^{\mathtt{D}}, \mathbf{d}_2^{\mathtt{D}}, ..., \mathbf{d}_{k^{\mathtt{D}}}^{\mathtt{D}}) \ .$$

La longueur des os associés à chaque degré de liberté est aussi organisé en un vecteur

$$\mathbf{l}^{\mathtt{D}} = (\mathbf{l}_1^{\mathtt{D}}, \mathbf{l}_2^{\mathtt{D}}, ..., \mathbf{l}_{k^{\mathtt{D}}}^{\mathtt{D}}) \ .$$

Dans cette thèse, on suppose que les valeurs angulaires sont toujours spécifiées en radians.

Une pose $\mathbf{d}^{\mathtt{D}}$ donnée permet de calculer et d'associer un **référentiel local** à chacun des joints. Le produit des matrices correspondant à ces référentiels le long de la hiérarchie du squelette permet d'obtenir un **référentiel global** pour chaque joint. La composante en translation

d'un référentiel global est la **position du joint**. On peut construire un vecteur en mettant bout à bout la position des différents joints

$$\mathbf{d}^{\mathtt{P}} = (\mathbf{d}_1^{\mathtt{P}}, \mathbf{d}_2^{\mathtt{P}}, ..., \mathbf{d}_{k^{\mathtt{P}}}^{\mathtt{D}}) \ ,$$

où $(\mathbf{d}_1^{\mathtt{P}}, \mathbf{d}_2^{\mathtt{P}}, \mathbf{d}_3^{\mathtt{P}})$ est la position de la racine, $(\mathbf{d}_4^{\mathtt{P}}, \mathbf{d}_5^{\mathtt{P}}, \mathbf{d}_6^{\mathtt{P}})$ est la position du premier joint, etc. La longueur des os associés à chaque joint est aussi organisée en un vecteur

$$\mathbf{l}^{\mathtt{P}} = (\mathbf{l}_1^{\mathtt{P}}, \mathbf{l}_2^{\mathtt{P}}, ..., \mathbf{l}_{k^{\mathtt{P}}}^{\mathtt{P}}) \ .$$

Notez que, pour i entier, $\mathbf{l}_{3i+1}^{\mathtt{P}} = \mathbf{l}_{3i+2}^{\mathtt{P}} = \mathbf{l}_{3i+3}^{\mathtt{P}}$, car un joint correspond à trois éléments dans le vecteur $\mathbf{d}^{\mathtt{P}}$ (la position en $\mathbf{x}$, $\mathbf{y}$ et $\mathbf{z}$).

En général, à chaque vecteur $\mathbf{d}^{\mathtt{P}}$ correspond une seule configuration des degrés de liberté $\mathbf{d}^{\mathtt{D}}$, qui peut être obtenue par une technique de **cinématique inverse globale**.[1] Une pose peut donc être décrite par la position de ses joints, $\mathbf{d}^{\mathtt{P}}$. De la même façon, on peut aussi décrire une pose par la position des marqueurs habillant un acteur lors d'une séance de capture de mouvements.

Le concept de pose décrit précédemment permet de distinguer deux configurations qui ne différeraient que par la direction dans laquelle le squelette fait face. Dans certaines applications, cependant, nous souhaitons considérer ces deux configurations comme identiques. Dans un tel cas, nous utilisons une **pose indépendante**. Celle-ci est décrite à l'aide d'un vecteur $\mathbf{d}^{\mathtt{I}}$ correspondant au vecteur $\mathbf{d}^{\mathtt{D}}$ modifié pour le rendre indépendant de l'orientation azimutale de la racine. On peut aussi construire $\mathbf{d}^{\mathtt{I}}$ pour le rendre indépendant de la position horizontale de la racine. Un tel vecteur $\mathbf{d}^{\mathtt{I}}$ est introduit au chapitre 5.

2.2.2 Signaux

Dans cette thèse, nous faisons souvent référence au concept de signal. La définition que nous adoptons pour ce terme est moins générale que la définition usuelle dans d'autres domaines.

Ainsi, par **signal scalaire**, nous entendons une série ordonnée de scalaires correspondant à des données échantillonnées à intervalles de temps réguliers. Un **signal vectoriel** est défini de la même façon avec des vecteurs. On note le signal scalaire s avec la série $s(1), s(2), ...$ et le signal vectoriel $\mathbf{s}$ par $\mathbf{s}(1), \mathbf{s}(2), ...$. La i-ième composante du signal vectoriel $\mathbf{s}$ est un signal scalaire et est notée $\mathbf{s}_i(1), \mathbf{s}_i(2), ...$. Dans un contexte où la nature du signal est claire, on omet les termes scalaire ou vectoriel.

[1] Certaines configurations extrêmes de $\mathbf{d}^{\mathtt{P}}$ peuvent être obtenues avec différents vecteurs $\mathbf{d}^{\mathtt{D}}$, mais en pratique ces cas sont rares.

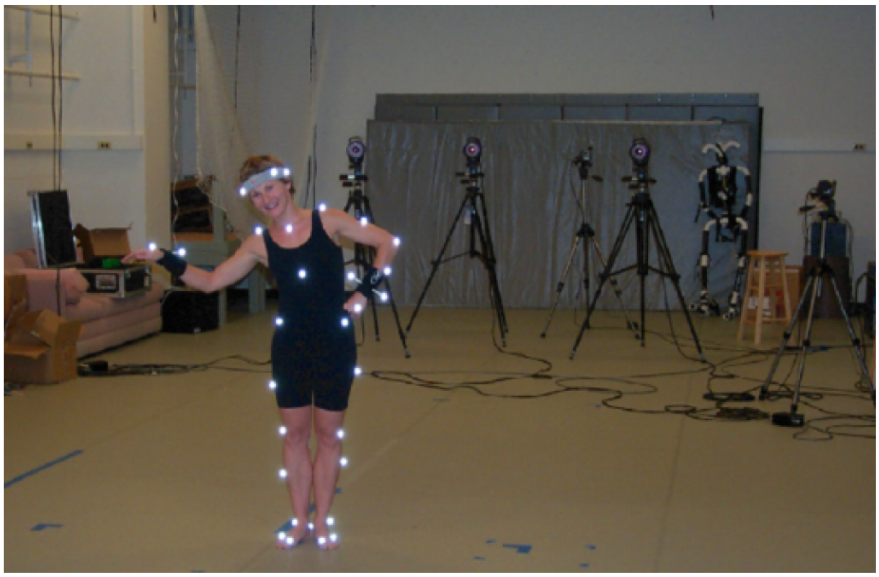


FIG. 2.2 – Acteur vêtu des marqueurs de capture de mouvements. Image tirée de [LCR+02].

On indexe parfois un signal avec une variable continue. Dans un tel cas la valeur $s(t)$ s'obtient en interpolant les échantillons à proximité de t.

2.2.3 Animations et mouvements

Le processus de **capture de mouvements** consiste à échantillonner régulièrement les positions d'un ensemble de marqueurs placés sur un acteur et qui permettent de reconstruire un signal $\mathbf{d}^{\mathsf{D}}$ (voir figure 2.2). Une **animation squelettique** ou une **animation acquise par capture de mouvements** est un signal $\mathbf{d}^{\mathsf{D}}$ correspondant à une série de gestes réalisés par l'acteur sans interruption. Lorsque le contexte est clair, nous utilisons simplement le terme **animation**. Certains travaux définissent une animation comme un signal correspondant aux trajectoires des marqueurs de capture de mouvements. Nous utilisons parfois cette définition lors de la revue des travaux antérieurs du chapitre 3.

Lors d'une séance de capture de mouvements typique, la capture sera interrompue régulièrement, ce qui donnera lieu à un ensemble d'animations. Une **banque de données d'animations** est le signal obtenu en juxtaposant ces animations.

Une **sous-séquence**, un **mouvement** ou un **geste** est une séquence d'échantillons représentant une partie d'une animation. On utilise généralement les termes mouvement et geste pour décrire une sous-séquence identifiable par un humain, comme par exemple un saut, un

cycle de marche ou un coup de poing.

Un **groupe de mouvements** est un concept que nous introduisons au chapitre 5 et qui correspond à un ensemble de sous-séquences représentant toutes un même geste.

2.3 Métriques de distorsion

La compression avec pertes d'une banque de données ou d'une animation $\mathbf{d}^{\mathtt{D}}$ introduit des distorsions et mène au signal approximatif $\hat{\mathbf{d}}^{\mathtt{D}}$. Pour déterminer l'efficacité d'une compression particulière, il est nécessaire d'évaluer le niveau de distorsion qu'elle introduit. Cette section s'intéresse à différentes façons d'évaluer ces distorsions.

2.3.1 Distorsion angulaire

La métrique de distorsion la plus simple est l'erreur RMS (*root mean square*) sur les degrés de liberté, que nous nommons aussi distorsion angulaire, et qui se définit par,

$$\varepsilon^{\mathtt{A}} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left|\mathbf{d}^{\mathtt{D}}(t) - \hat{\mathbf{d}}^{\mathtt{D}}(t)\right|^2},$$

où n est le nombre d'échantillons du signal.

2.3.2 Distorsion angulaire pondérée

La distorsion angulaire suppose une importance uniforme pour tous les degrés de liberté. Or $\mathbf{d}_1^{\mathtt{D}}, \mathbf{d}_2^{\mathtt{D}}$ et $\mathbf{d}_3^{\mathtt{D}}$ représentent des positions alors que tous les autres degrés de liberté représentent des angles. De plus, en raison de l'organisation hiérarchique des joints, certains degrés de liberté ont une influence sur tout le squelette alors que d'autres n'affectent qu'un petit nombre de joints. Pour pallier à ce problème, nous introduisons la distorsion angulaire pondérée,

$$\varepsilon^{\mathtt{D}} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\sum_{i=1}^{k^{\mathtt{D}}}\mathbf{w}_i\left(\mathbf{d}_i^{\mathtt{D}}(t) - \hat{\mathbf{d}}_i^{\mathtt{D}}(t)\right)^2},$$

dans laquelle on a introduit une pondération $\mathbf{w}_i$ indépendante pour chaque degré de liberté.

Cette dernière distorsion exige que nous définissions le vecteur de pondérations $\mathbf{w}$. Nous avons opté pour un schéma de pondération qui tente d'évaluer l'importance relative de chaque joint tout en s'assurant d'uniformiser les unités. Ainsi nous multiplions les angles (en radians) par des longueurs (en cm) de façon à les rendre comparables aux positions contenues dans $\mathbf{d}_1^{\mathtt{D}}, \mathbf{d}_2^{\mathtt{D}}$ et $\mathbf{d}_3^{\mathtt{D}}$.

On définit tout d'abord $\mathbf{w}_1 = \mathbf{w}_2 = \mathbf{w}_3 = 1$. Pour $i > 3$, on pose $\mathbf{w}_i$ comme étant la longueur de la plus grande chaîne d'os influencée par $\mathbf{d}_i^{\mathtt{D}}$. De cette façon, les joints à la racine de la hiérarchie auront une importance plus grande que les joints près des feuilles. Cette pondération reste cependant très approximative. Par exemple, si le bras gauche est plié et que le bras droit est étendu, les angles de l'épaule gauche ont une importance moins grande que ceux de l'épaule droite sur l'apparence finale du squelette.

2.3.3 Distorsion positionnelle

Bien qu'en pratique la distorsion $\varepsilon^{\mathtt{D}}$ s'avère plus utile que $\varepsilon^{\mathtt{A}}$, il n'en reste pas moins que la définition de $\mathbf{w}$ est plutôt *ad hoc*. Une façon plus formelle d'évaluer la distorsion engendrée par une compression serait d'étudier l'effet de cette compression sur la position des joints. À cette fin nous introduisons la distorsion positionnelle,

$$\varepsilon^{\mathtt{P}} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\sum_{i=1}^{k^{\mathtt{P}}} \frac{\mathbf{l}_i^{\mathtt{P}}}{\sum_{j=1}^{k^{\mathtt{P}}}\mathbf{l}_j^{\mathtt{P}}}\left(\mathbf{d}_i^{\mathtt{P}}(t) - \hat{\mathbf{d}}_i^{\mathtt{P}}(t)\right)^2}.$$

Dans cette équation, on introduit la pondération $\mathbf{l}_i^{\mathtt{P}} / \sum_{j=1}^{k^{\mathtt{P}}} \mathbf{l}_j^{\mathtt{P}}$ qui fait en sorte que les joints associés à des os plus longs auront une influence plus importante.

L'avantage principal de la distorsion positionnelle est qu'elle mesure bien l'effet d'une erreur sur la configuration du squelette dans l'espace. Ainsi, elle s'avère une mesure plus précise que $\varepsilon^{\mathtt{A}}$ ou $\varepsilon^{\mathtt{D}}$ pour évaluer la qualité d'une compression avec pertes telle qu'elle sera perçue par un observateur.

La distorsion positionnelle a cependant le désavantage de requérir une transformation depuis les degrés de liberté $\mathbf{d}^{\mathtt{D}}$ vers la position des joints $\mathbf{d}^{\mathtt{P}}$. Comme nous le verrons au chapitre 4, cette transformation non-linéaire peut complexifier grandement le processus d'optimisation lors de la compression.

2.3.4 Évaluation visuelle

Dans le contexte d'une animation destinée à être observée par un individu, une mesure de distorsion idéale devrait être dépendante de la qualité perceptuelle de l'animation compressée et, ainsi, devrait prendre en considération les limites connues de la vision humaine. Il va sans dire que les métriques de distorsion introduites précédemment ne prétendent pas accomplir cette tâche. En fait, l'évaluation quantitative de la qualité perceptuelle d'une animation est un problème complexe et vaste que nous n'abordons pas dans cette thèse.

Néanmoins, nous avons adopté une approche utilisée par différents chercheurs dans le domaine et qui consiste à évaluer visuellement les animations. Pour ce faire, nous affichons à l'écran une représentation 3D simple du squelette animé et observons l'animation de plusieurs points de vue différents. Aussi, pour permettre d'illustrer les résultats sous forme imprimée, nous superposons l'animation originale et l'animation compressée, comme par exemple à la figure 4.12. Nous invitons aussi le lecteur à visualiser les séquences vidéo accompagnant cette thèse et disponibles sur internet [Bea07a, Bea07b, Bea07c].

Chapitre 3

Travaux antérieurs

It has long been the dearest wish of Mr. Norrell's heart to bring a more precise understanding of modern magic before a wider audience, but alas, sir, private wishes are often frustrated by public duties.

Mr Lascelles

La compression de données est un domaine vaste qui trouve de nombreuses sphères d'application. En se limitant uniquement aux applications reliées à l'infographie, on découvre que la compression est utilisée pour l'encodage d'images [Kou95, Wal91], de séquences vidéo [Sym03, 14491a, 14491b], de mémoires de trame [BP04], de maillages [Ros99], etc. Plus récemment, certains chercheurs ont proposé d'étudier les façons d'utiliser la compression pour stocker des animations acquises par capture de mouvements [LZWM05, LM06, Ari06]. Nous consacrons donc ce chapitre à la revue de ces travaux mais, surtout, nous présentons les principaux résultats du large domaine dans lequel ils s'inscrivent : l'animation de personnages à partir de données (*data-driven character animation*).

En vue d'asseoir les principales notions théoriques utilisées dans cette thèse, ce chapitre propose dans un premier temps une brève revue de certains concepts fondamentaux en compression de signaux temporels. Nous abordons par la suite une application de ces concepts en infographie 3D : la compression de maillages animés. Le reste du chapitre est dédié aux travaux en analyse d'animations puis aux recherches antérieures en compression de données acquises par capture de mouvements.

3.1 Compression de signaux temporels

Il est possible d'appliquer une compression avec pertes à de nombreux types de signaux différents. Pour les signaux sonores, et particulièrement pour la musique, le format MP3 et ses variantes ont connu un engouement important ces dernières années. Plus spécifiquement, cette compression s'inscrit dans la standard MPEG-4 [14491a, 14491b] qui décrit les différentes facettes de la compression de signaux d'animation audio-vidéo. D'autres standards s'attardent plutôt aux techniques de compression de la voix, souvent adaptées au faible débit des appareils de communication cellulaires [RWO98]. Les chercheurs se sont aussi intéressés à la compression de signaux moins communs, comme les animations de maillages 3D que nous abordons en détail dans la prochaine section.

La nature des types de signaux énumérés précédemment diffère passablement : une pièce de musique classique numérisée ne présente pas les mêmes structures que la voix humaine qui à son tour se distingue clairement d'une animation vidéo ou de celle d'un maillage 3D. Ces signaux partagent cependant tous une caractéristique commune : ils dépendent du temps.

L'omniprésence des signaux temporels a engendré un intérêt important pour les recherches dans ce domaine qui ont à leur tour mené au développement de plusieurs approches générales aujourd'hui bien connues et bien étudiées. Celles-ci touchent à de nombreux champs d'étude qu'il serait trop long de couvrir en détail dans cette thèse. Le lecteur souhaitant approfondir ses connaissances dans ces différents domaines est invité à consulter un des nombreux ouvrages de référence disponibles. Par exemple [Say96, Dro02, Sal02, Sal04] présentent en détail les bases générales de la compression de données tandis que [SDS96, PI97, Mal98, Uyt99, BBN00] abordent les notions spécifiquement reliées aux transformées en ondelettes.

Le reste de cette section s'attarde brièvement à trois aspects de ces domaines plus spécifiquement reliés à cette thèse et dont nous ferons un usage important par la suite. Dans un premier temps, nous proposons un survol de la technique de compression générique qu'est l'encodage par transformation. Ensuite, nous présentons deux types de transformations : l'analyse des composantes principales (*PCA*) et la transformation en ondelettes.

3.1.1 Encodage par transformation

L'encodage par transformation [Sal04] peut être utilisé sur n'importe quelle donnée vectorielle. La technique consiste simplement à faire subir une transformation au vecteur et à l'encoder dans sa nouvelle représentation. Si on se limite aux transformations linéaires, comme c'est

le cas dans cette thèse, alors le vecteur $\mathbf{v}$ est transformé en $\mathbf{v}' = \mathbf{M}\mathbf{v}$ qui sera encodé par la suite. Lorsque $\mathbf{M}$ est une matrice inversible, alors le décodage se fait facilement en utilisant $\mathbf{v} = \mathbf{M}^{-1}\mathbf{v}'$.

Si $\mathbf{M}$ est inversible, $\mathbf{v}'$ a le même nombre de dimensions que $\mathbf{v}$ et rien n'indique que l'encodage par transformation ne permette de compresser les données. Cependant, si l'ensemble des vecteurs à encoder se trouve sur un sous-espace linéaire de dimension inférieure, il est possible de trouver une transformation $\mathbf{M}$ qui annule certaines des composantes de $\mathbf{v}'$. Par exemple, imaginons que les vecteurs à encoder $\mathbf{v} \in \mathbb{R}^3$ soient tous le long de la droite passant par l'origine et par $(1, 1, 1)$. Prenons la matrice

$$\mathbf{M} = \begin{bmatrix} 1 & 1 & 1 \\ -2 & 1 & 1 \\ 0 & -1 & 1 \end{bmatrix}.$$

On peut alors facilement montrer que tous les $\mathbf{v}' = \mathbf{M}\mathbf{v}$ sont de la forme $(k, 0, 0)^\mathsf{T}$ et qu'ils peuvent donc être encodés à l'aide d'un seul coefficient scalaire.

En général, les vecteurs à encoder ne s'inscrivent pas parfaitement dans un sous-espace linéaire. Cependant, on peut souvent trouver une transformation linéaire qui fasse en sorte que beaucoup de coefficients prennent des valeurs près de zéro. On dit d'une telle transformation qu'elle compacte l'énergie du vecteur dans ses premiers coefficients. Un exemple d'une telle transformation, pour des vecteurs 2D, est donné à la figure 3.1.

L'intérêt d'une telle transformation apparaît lorsqu'on souhaite effectuer une compression avec pertes grâce à une quantification (*quantization*). Les coefficients proches de zéro peuvent alors être quantifiés dans un faible nombre de cases (*bins*) pouvant s'exprimer sur peu de bits. On peut même tronquer complètement certains coefficients trop petits et réduire d'autant la taille de l'encodage. C'est cette dernière stratégie que nous utilisons principalement dans le cadre de ce mémoire.

Un encodage par transformation peut être utilisé quand les signaux sont des vecteurs variant dans le temps. Dans un tel cas, chaque échantillon est un vecteur qui peut être transformé puis encodé. Il est aussi possible d'utiliser l'encodage par transformation sur un signal temporel à une seule dimension. Pour ce faire, il suffit de construire un vecteur composé d'une série d'échantillons successifs.

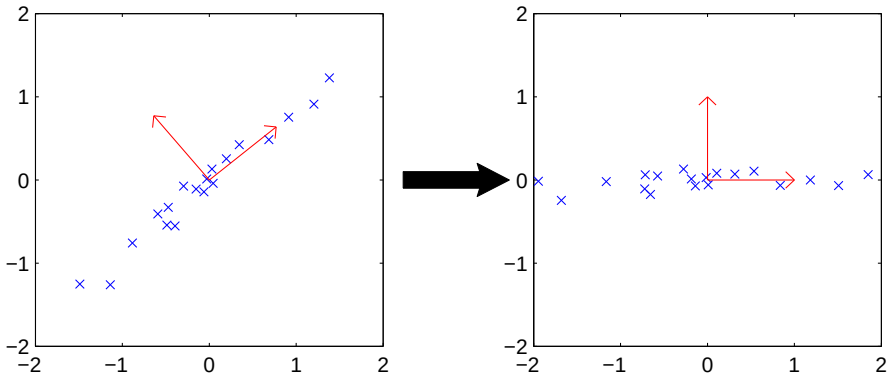


FIG. 3.1 – Exemple d'un encodage par transformation. À gauche, on doit encoder 20 vecteurs 2D, l'encodage direct des vecteurs nécessite d'utiliser la même quantification en $\mathbf{x}$ qu'en $\mathbf{y}$. À droite, les données ont subi une transformation linéaire et les vecteurs varient maintenant surtout selon l'axe $\mathbf{x}$. On pourrait ainsi utiliser une quantification beaucoup plus précise en $\mathbf{x}$ qu'en $\mathbf{y}$, voire laisser tomber entièrement la coordonnée $\mathbf{y}$. Notez que dans cet exemple, la nouvelle base a été déterminée par *PCA*.

3.1.2 Analyse des composantes principales

Une des transformations qui peut être utilisée avec l'encodage précédent est l'analyse des composantes principales, ou *PCA* [Jol86]. La *PCA* est une transformation linéaire orthogonale, c'est-à-dire qu'elle peut s'exprimer avec une matrice où les colonnes sont orthogonales et normalisées. Celle-ci s'obtient simplement en effectuant la décomposition en valeurs singulières d'une matrice formée par les vecteurs à encoder.

La matrice *PCA* dépend donc de l'ensemble des vecteurs qui composent le signal. La première colonne de la matrice, soit le premier vecteur de base, correspond à la direction selon laquelle le signal a une variance maximale. Les autres colonnes sont ensuite ordonnées de façon à ce que le dernier vecteur de base représente la direction de variance minimale.

Ainsi, si les vecteurs à encoder sont proches d'un sous-espace linéaire, les derniers coefficients de la représentation *PCA* seront pratiquement nuls. En fait, plus la dimension de ce sous-espace est faible, plus le nombre de coefficients proches de zéro sera élevé. On peut démontrer qu'une projection limitée aux premiers vecteurs de la base *PCA* est optimale au sens des moindres carrés. Un exemple d'utilisation de la *PCA* en 2D est donné à la figure 3.1.

3.1.3 Transformée en ondelettes

Si l'analyse des composantes principales permet d'identifier un sous-espace linéaire optimal pour la projection, elle comporte cependant un désavantage important. En effet, la base de projection obtenue par *PCA* dépend du signal. Il est donc nécessaire d'identifier un ensemble de vecteurs de base différent pour chaque groupe de signaux à encoder. Lorsque la dimensionnalité du signal est importante, la taille de ces vecteurs est elle aussi importante, engendrant un coût additionnel (*overhead*) significatif dans l'encodage et réduisant d'autant le taux de compression.

Pour bien comprendre ce problème considérons l'exemple d'un encodage par transformation *PCA* de 100 vecteurs correspondant chacun à des signaux de 256 échantillons. Si on projette ces signaux sur les 10 dimensions principales, alors ils pourront être encodés en utilisant un total de $100 \times 10 = 1000$ coefficients. En revanche, le stockage de la base demandera 10 vecteurs de 256 éléments pour un total de 2560 scalaires. Dans un tel cas, l'en-tête requiert plus d'espace que les signaux compressés eux-mêmes !

Pour contourner ce problème, on pourrait utiliser une base moins optimale mais assez générale pour permettre un encodage efficace de tous les signaux. Il est facile de montrer que, si les signaux sont totalement arbitraires, il n'existe aucune base générale satisfaisante. Heureusement les signaux temporels obtenus par capture de mouvements possèdent un certain nombre de caractéristiques qui nous permettent d'espérer trouver une base adaptée à leur représentation.

En particulier ces signaux sont continus, assez lisses et oscillent souvent à proximité de certaines valeurs. On peut donc espérer les représenter de façon compacte dans une base partageant ces différentes caractéristiques. Les bases de type Fourier[1], dont les fonctions de base sont périodiques et lisses, semblent particulièrement adaptées à cette tâche. Ces fonctions sont en fait des sinusoïdes à support non borné. Chaque coefficient contrôle donc l'importance d'une fréquence particulière sur la totalité des échantillons. Ainsi, ces bases sont beaucoup plus aptes à représente de façon compacte un signal dont les caractéristiques fréquentielles ne varient pas en fonction du temps. La figure 3.2 montre que ce n'est malheureusement pas le cas des signaux qui nous occupent. On y distingue en effet une partie comprenant des oscillations à hautes fréquences alors que le reste du signal varie beaucoup moins brusquement.

La théorie des ondelettes nous laisse contourner cette limitation en permettant la création de fonctions de base à support fini partageant beaucoup des caractéristiques de la base de Fourier.

[1] Par bases de type Fourier, on entend les bases définies implicitement par la transformée de Fourier discrète ou, plus communément en compression, par la transformée en cosinus discrète (*DCT*).

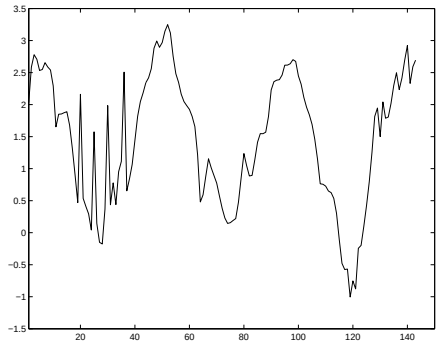


FIG. 3.2 – Signal d'animation correspondant à l'évolution temporelle de l'angle de rotation par rapport à $\mathbf{x}$ du joint *upperback* pendant une animation de course. On remarque que les caractéristiques fréquentielles ne sont pas constantes : des oscillations rapides se produisent au début mais le reste de la courbe est assez lisse.

Comme les fonctions ont un support fini, elles sont en mesure d'exprimer de hautes fréquences localement sans affecter le reste du signal.

La théorie des ondelettes est un sujet vaste qui peut être approché formellement dans une optique fréquentielle via l'analyse de Fourier, comme l'ont fait les pionniers du domaine [Chu92a, Chu92b, She92, Dau92, CDF92]. Nous préférons cependant l'approche plus intuitive proposée par Sweldens [Swe96] et où les bases d'ondelettes sont construites entièrement dans le domaine spatial. La technique, baptisée schéma de *lifting* (*lifting scheme*) par Sweldens, comprend un certain nombre d'opérations simples qui, lorsqu'elles sont appliquées récursivement, permettent de transformer les n échantillons d'un signal en une série de n coefficients d'ondelettes. La forme et les caractéristiques de la base d'ondelettes résultante dépendent des opérations choisies et de l'ordre dans lequel elles ont été appliquées.

Dans le cadre de cette thèse nous utilisons trois opérations de *lifting* : la décimation (*split*), la prédiction et la mise à jour. La décimation consiste à diviser un signal pour en isoler les échantillons pairs d'une part et impairs de l'autre (figure 3.3(a)). La prédiction consiste à effectuer une combinaison linéaire des échantillons d'un signal, puis à soustraire le résultat d'un second signal (figure 3.3(b)). La combinaison linéaire choisie vise à prédire le contenu du second signal à partir du premier et est donc généralement composée de coefficients d'interpolation. La mise à jour est similaire mais consiste à additionner au second signal le résultat d'une combinaison linéaire des échantillons du premier signal (figure 3.3(c)). Le but de l'étape de mise à jour est de façonner la base d'ondelettes pour qu'elle possède un certain nombre de caractéristiques,

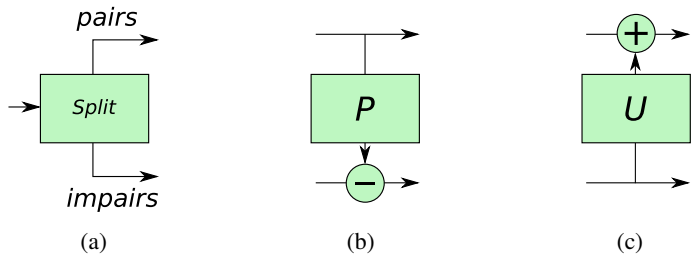


FIG. 3.3 – Opérations de *lifting* direct : (a) la décimation, (b) la prédiction, (c) la mise à jour.

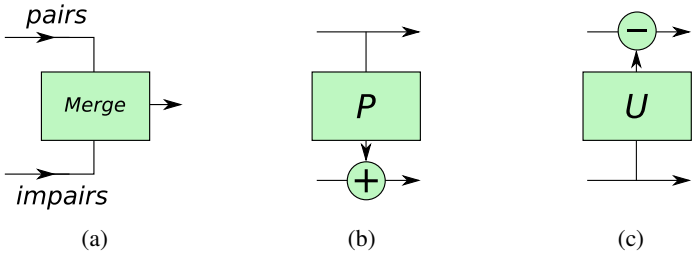


FIG. 3.4 – Opérations de *lifting* inverse : (a) la fusion (décimation inverse), (b) la prédiction inverse, (c) la mise à jour inverse.

comme par exemple des propriétés de symétrie ou de continuité.

Sweldens et Schröder [SS95] expliquent de façon pratique comment ces opérations peuvent être agencées pour concevoir des algorithmes de transformation en ondelettes directe. Les opérations peuvent être facilement inversées (figure 3.4) pour permettre l'élaboration des algorithmes de transformation en ondelettes inverse. Les algorithmes résultants peuvent être implantés sans allocation de mémoire (*in-place*) et requièrent un nombre d'opérations de l'ordre du nombre d'échantillons dans le signal, $O(n)$. Ils sont donc généralement très efficaces.

Sweldens [Swe96] montre que, lorsque le schéma de *lifting* est utilisé sur un signal périodique, les ondelettes obtenues peuvent s'exprimer par des translations et des mises à l'échelle d'une ondelette mère. Il montre aussi que ces ondelettes sont identiques aux ondelettes biorthogonales pouvant être produites par les approches classiques [CDF92].

Le schéma de *lifting* offre cependant un avantage important en permettant la création d'ondelettes dites de *seconde génération* [Swe98] qui lèvent un certain nombre de restrictions des ondelettes biorthogonales classiques. Il devient ainsi possible de définir la transformation en ondelettes d'un signal échantillonné de façon irrégulière, ou encore d'un signal apériodique défini sur un intervalle. Dans ce dernier cas, la base d'ondelettes obtenue s'ajuste aux frontières de l'intervalle et ne s'exprime donc plus comme de simples translations ou mises à l'échelle d'une

ondelette mère. Étant donné que les signaux que nous traitons ne sont pas périodiques, nous utilisons cette dernière particularité dans le cadre de cette thèse. Notez que les vecteurs de la base d'ondelettes ne dépendent pas des données elle-mêmes. Donc, contrairement à la *PCA*, le problème du stockage de la base ne se pose pas.

3.2 Compression de maillages animés

Les techniques décrites précédemment sont des outils généraux qui peuvent être utilisés de nombreuses façons. En infographie 3D, un des exemples les plus importants de recherche en compression de signaux temporels se trouve dans les travaux en compression de maillages animés.

L'intérêt des recherches en compression de maillages animés s'explique par la taille très importante requise pour leur stockage direct. En effet, dans leur forme brute ces données sont composées d'une position 3D par sommet pour chaque instant échantillonné. Comme le nombre de sommets est en général très important (certains maillages en comptent plusieurs millions), on comprend facilement la nécessité d'une compression. Il est possible de considérer une animation comme une séquence de maillages statiques qui peuvent être compressés indépendamment par une des nombreuses techniques développées à cet effet (*e.g.* [Ros99]). On peut cependant augmenter de façon importante le taux de compression en s'intéressant à la cohérence temporelle présente dans une telle séquence.

Cette section offre un bref aperçu des techniques de compression de maillages animés. Les techniques présentées exigent que tous les maillages de la séquence soient isomorphes. Ainsi, la topologie n'est en général pas compressée et est transmise en en-tête de l'animation. La compression de maillages non isomorphes est un problème beaucoup plus rarement étudié [YKL04].

La compression de maillages animés reste un problème passablement différent de la compression de données acquises par capture de mouvements. Cet aperçu nous permet néanmoins d'identifier un certain nombre de pistes intéressantes quant aux applications potentielles de nos travaux.

3.2.1 Segmentation et prédiction

Plusieurs techniques procèdent tout d'abord à la segmentation du maillage en sous-régions subissant des déformations plus ou moins similaires lors de l'animation. À chaque instant, une

FIG. 3.5 – Méthode de segmentation de maillage animé de James et Twigg [JT05]. Image tirée de [JT05].

déformation optimale est déterminée et est utilisée pour prédire la position des sommets de la sous-région. La position véritable des sommets est soustraite de leur position prédite et seule cette erreur résiduelle est encodée. Comme l'erreur est généralement proche de zéro, un encodage par quantification permet de réduire sensiblement la taille des données. Les différentes techniques décrites ci-après sont toutes basées sur ce schéma général mais utilisent différentes segmentations ou différents modèles de déformations.

La première technique du genre a été introduite par Lengyel [Len99]. Il propose de segmenter le maillage de façon à maximiser la similarité des mouvements subis par les sommets d'une même sous-région. Le déplacement d'une sous-région est ensuite prédit à l'aide d'une matrice de transformation rigide. Sattler *et al.* [SSK05] utilisent la technique des *PCA* agrégées (*clustered PCA*) pour séparer le maillage en régions dont l'animation est presque indépendante. Ils compressent ensuite chacune des régions ainsi obtenues par *PCA*.

Bien qu'ils ne visent pas directement la compression de maillages, James et Twigg [JT05] proposent une technique similaire à celle de Lengyel. Dans leur approche, les sous-régions sont libres de subir des transformations non rigides quelconques. De plus, un sommet peut appartenir à plus d'une sous-région. La transformation subie par le sommet est une somme pondérée des transformations associées aux différentes régions auxquelles il appartient. Ces pondérations sont obtenues automatiquement par un mécanisme d'optimisation. La technique est illustrée à la figure 3.5.

Ahn *et al.* [AKKH01] utilisent une segmentation beaucoup plus simple en séparant le maillage en bandes de triangles (*triangle strips*) de longueur constante. Le mouvement de chaque sous-région ainsi obtenue est modélisée par un unique vecteur de mouvement. Finalement, un encodage par transformation *DCT* est appliqué à l'erreur résiduelle.

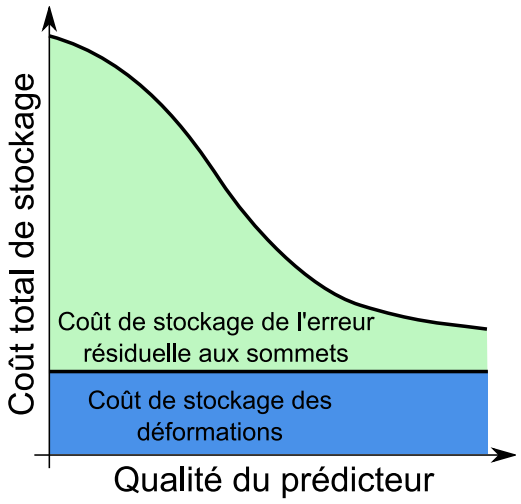


FIG. 3.6 – La compression d'un maillage animé demande de stocker l'erreur résiduelle aux sommets après prédiction ainsi que les déformations utilisées par le prédicteur. Cette figure montre que l'importance relative du coût de stockage des déformations augmente avec la qualité du prédicteur. Pour un prédicteur de bonne qualité, il peut devenir important de compresser aussi les déformations. (La courbe illustrée n'est qu'indicative et sa forme est arbitraire.)

Il existe un parallèle important entre les techniques précédentes et la compression de données acquises par capture de mouvements. En effet, les déformations obtenues pour chaque sous-région peuvent souvent être exprimées avec un nombre limité de degrés de liberté. Dans certaines approches [JT05, SSK05] ces degrés de liberté peuvent même être organisés de façon hiérarchique comme c'est le cas pour les données acquises par capture de mouvements. La compression de l'évolution temporelle des déformations se rapproche ainsi passablement du problème abordé dans cette thèse.

Malheureusement, aucun des auteurs précédents ne s'intéresse à la compression des déformations. Ceci s'explique probablement par le fait que le stockage de ces déformations requiert en général beaucoup moins d'espace mémoire que le stockage de l'erreur résiduelle aux sommets. Ce n'est cependant vrai que dans la mesure où le modèle de déformation constitue un prédicteur imprécis des déformations réelles subies par le maillage. Dans le cas, par exemple, où la technique de James et Twigg [JT05] est appliquée à un maillage animé par *skinning* [LCF00], le prédicteur est très précis et l'erreur résiduelle est presque nulle. Le coût relatif du stockage des déformations augmente alors de façon importante. Nous illustrons ce phénomène graphiquement à la figure 3.6. Une telle situation constitue une sphère d'application potentielle pour

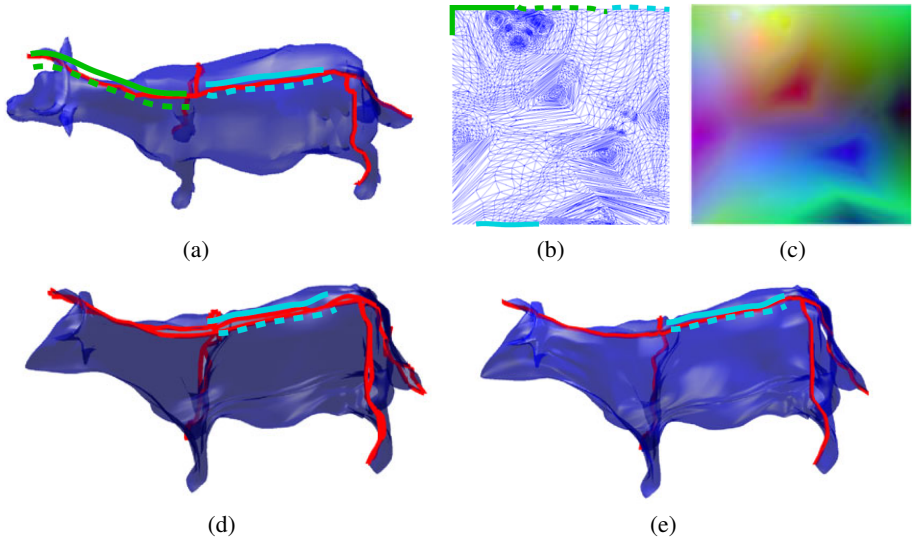


FIG. 3.7 – Technique des séquences vidéo géométriques de Briceño *et al.* [BSM+03] (a) Chaque maillage de la séquence est découpé de façon à devenir (b) isomorphe à un plan. (c) La géométrie est échantillonnée régulièrement dans ce plan pour produire une séquence d'images géométriques [GGH02] qui peut ensuite être compressée par des techniques semblables à MPEG. (d) Le maillage est reconstruit puis (e) les erreurs introduites aux coupures sont corrigées. Image tirée de [BSM+03].

les techniques introduites dans cette thèse.

3.2.2 Autres approches

Des auteurs ont proposé des approches utilisant le codage par transformation. En effet, chaque échantillon correspond à un maillage statique qui peut s'exprimer sous la forme d'un unique vecteur de très haute dimensionnalité. Alexa et Müller [AM00] utilisent la *PCA* pour trouver un petit nombre de maillages de base dont la combinaison linéaire permet de reproduire l'animation assez fidèlement. Ces travaux ont été étendus par Karni et Gotsman [KG04a] qui introduisent un prédicteur linéaire pour les coefficients de la *PCA*. Guskov et Khodakovsky [GK04] appliquent quant à eux une transformation en ondelettes au vecteur correspondant à chaque maillage. Une telle transformation peut être définie sur un maillage quelconque en utilisant des ondelettes de seconde génération [Swe98]. Les coefficients d'ondelettes de chaque maillage montrent une cohérence temporelle importante et sont encodés en utilisant un schéma de prédiction simple.

Briceño *et al.* [BSM+03] étendent la technique des images géométriques (*geometry images*) de Gu *et al.* [GGH02] pour produire des séquences vidéo géométriques (*geometry videos*). L'approche consiste à identifier tout d'abord une coupe de la topologie permettant de la rendre isomorphe à un plan et produisant une paramétrisation 2D globale. Cette paramétrisation est utilisée pour rééchantillonner régulièrement la géométrie de tous les maillages de la séquence. Les maillages originaux sont remplacés par les versions rééchantillonnées qui peuvent s'encoder sous la forme d'une structure 2D régulière nommée images géométriques. Une séquence d'images géométriques peut être compressée avec des techniques semblables à celles utilisées pour la compression de séquences vidéo comme MPEG [14491a, 14491b]. Briceño *et al.* compressent certaines images géométriques de façon indépendante avec une base en ondelettes 2D. Ils utilisent aussi une transformation linéaire globale comme mécanisme de prédiction pour exploiter la cohérence temporelle. La technique est illustrée à la figure 3.7.

Yang *et al.* [YKL02] évaluent un vecteur de mouvement pour chaque sommet du maillage animé et ce, à chaque échantillon. Ces vecteurs présentent des cohérences spatiales (pour des sommets rapprochés sur le maillage) ainsi que des cohérences temporelles. Les auteurs introduisent des prédicteurs exploitant ces deux types de cohérences.

Zhang et Owen [ZO04] divisent l'espace entourant chaque maillage en un *octree*. Un vecteur 3D de déplacement est associé à chaque sommet de l'*octree*. Une interpolation trilinéaire permet de prédire le déplacement subi par un sommet du maillage. Si l'erreur de prédiction est trop importante, l'*octree* est subdivisé et de nouveaux vecteurs de déplacement sont calculés.

Quant à la technique *dynapack*, développée par Ibarra et Rossignac [IR03], elle combine différents prédicteurs temporels au prédicteur spatial proposé par Touma et Gotsman [TG98] pour l'encodage d'un maillage statique.

3.3 Analyse d'animations

Bien que les travaux touchant spécifiquement à la compression de données acquises par capture de mouvements soient assez rares, beaucoup de chercheurs se sont intéressés au domaine plus vaste que constitue l'animation de personnages à partir de données (*data-driven character animation*). Les recherches dans ce domaine concernent deux facettes relativement orthogonales : l'analyse des données acquises par capture de mouvements et la synthèse d'animations. Dans cette section nous nous intéressons principalement aux méthodes d'analyse qui sont plus

susceptibles de nous fournir des pistes ou des outils pour le développement de techniques de compression.

Une méthode d'analyse consiste, dans sa définition la plus large, à séparer un problème en parties indépendantes plus faciles à traiter. L'analyse de données de mouvements peut donc être vue comme la recherche des structures présentes dans les données. Les formes d'analyse proposées dans la littérature sont de natures très variées et cherchent souvent à identifier des structures adaptées à une application spécifique. Notre revue tente plutôt de classifier les méthodes d'analyse en fonction de leur nature-même et non des applications visées. Le reste de cette section se consacre ainsi tour à tour aux méthodes d'analyse de signal, de segmentation, de recherche d'animations, de modélisation statistique, et de création de graphes de mouvements.

3.3.1 Analyse de signaux d'animation

Plusieurs techniques d'analyse d'animations sont fortement inspirées de méthodes de traitement de signaux classiques. Dans cette section, nous présentons deux classes de techniques de ce genre : la combinaison d'animations et l'analyse fréquentielle.

Combinaison d'animations

Dans certaines situations, on dispose de plusieurs animations représentant des exécutions différentes d'un même mouvement. La combinaison d'animations (*animation blending*) est une technique par laquelle ces animations peuvent être interpolées de façon à créer une nouvelle animation intermédiaire. Par exemple, un saut court et un saut long pourraient être combinés de façon à produire un saut moyen.

Perlin [Per95] ainsi que Witkin et Popović [WP95] ont proposé une technique simple basée sur une combinaison linéaire convexe des animations. Bruderlin et Williams [BW95] remarquent qu'il est souvent nécessaire de faire correspondre les moments clés entre les deux animations, comme par exemple l'instant où un pied touche le sol. Pour ce faire ils introduisent l'alignement temporel dynamique (voir section 3.3.3). Ce principe a plus tard été étendu par Kovar et Gleicher [KG03] puis par Ménardais *et al.* [MKMA04] pour la combinaison de plus de deux animations.

Certains auteurs ont aussi proposé d'utiliser une représentation alternative des animations avant de les combiner. Unuma *et al.* [UAT95] interpolent les animations dans le domaine de Fourier. Wiley et Hahn [WH97] utilisent une représentation alternative du squelette dans la-

quelle chaque référentiel peut se déplacer indépendamment des autres. Certains os du squelette peuvent ainsi s'allonger suite à une combinaison d'animations, mais les auteurs remarquent que cet effet est généralement négligeable. Rose *et al.* [RCB98] encodent chaque signal d'animation à l'aide de B-splines cubiques dont ils interpolent les points de contrôle. Pour éviter les problèmes de *gimbal lock* liés aux angles d'Euler, Park *et al.* [PSS02, PSS04] proposent d'utiliser une paramétrisation exponentielle [Gra98] lors de la combinaison d'animations. Abe *et al.* [ALP04], quant à eux, produisent des séquences physiquement plausibles en combinant les moments associés à chaque joint plutôt que les angles ou les positions.

Un problème lié à ce type de technique vient du fait que les paramètres de combinaison forment un espace abstrait relié de façon non linéaire aux objectifs usuels en synthèse d'animations. Par exemple, on pourrait souhaiter générer une animation où les mains atteignent des positions précises mais il n'est pas simple de déterminer quelle combinaison d'animations nous permettra de produire ce résultat. Certains auteurs [RSC01, KG04b, ALP04] ont proposé de contourner ce problème en sur-échantillonnant l'espace des paramètres des combinaisons. Mukai et Kuriyama [MK05], quant à eux, introduisent le concept d'interpolation géostatistique qui modélise statistiquement la relation entre les paramètres et les poses générées.

En terminant, il est intéressant de noter que Sofanova et Hodgins [SH05] ont montré que la simple combinaison d'animations pouvait souvent permettre de générer des animations physiquement valides. Ikemoto *et al.* [IAF07] montrent aussi comment la combinaison de deux à quatre animations peut être utilisée pour transitionner rapidement entre deux mouvements très différents.

Analyse fréquentielle

Les techniques de Fourier ont fréquemment été utilisées pour le traitement des signaux d'animation. Dans le système d'animation *Inkwell*, introduit par Litwinowicz [Lit91], l'utilisateur peut manipuler les paramètres d'un filtre IIR (*infinite impulse response filter*) de manière à affecter la fonction contrôlant l'évolution temporelle d'un point de contrôle. Unuma *et al.* [UAT95] appliquent une transformée de Fourier sur des séquences cycliques de marches et de courses et peuvent ensuite manipuler des paramètres tels la vitesse, la longueur d'un pas, etc. Bruderlin et Williams [BW95] utilisent une banque de filtres pour obtenir une représentation multi-résolution des animations (figure 3.8). Une technique similaire est adoptée par Pullen et Bregler [PB00, PB02] pour combiner un croquis d'animation grossier proposé par l'usager et

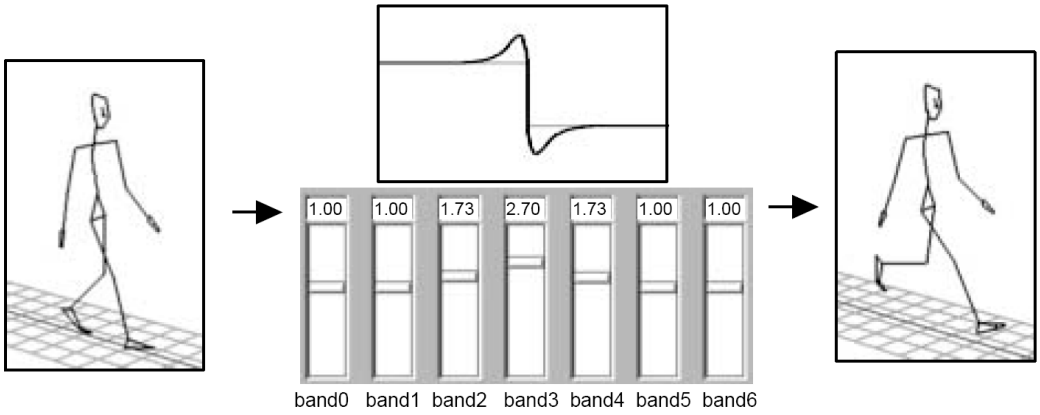


FIG. 3.8 – Bruderlin et Williams [BW95] proposent d'utiliser une banque de filtres pour contrôler le caractère d'une animation. Chaque glissière (*slider*) correspond à une gamme de fréquences dont le gain peut être ajusté indépendamment. Image tirée de [BW95].

des séquences précises obtenues par capture de mouvements.

Plusieurs auteurs [LGC94, LS99, KLS00, LS00, LS01, LS02] analysent et manipulent les signaux d'animation après les avoir transformés dans une base d'ondelettes. Une telle représentation est utilisée pour accélérer des algorithmes d'optimisation dynamique, faciliter l'édition du mouvement, filtrer les animations, ou leur appliquer des filigranes numériques (*watermark*).

3.3.2 Segmentation de mouvements

La capacité à segmenter automatiquement une animation en fragments plus courts s'avère souvent un outil de grande valeur en permettant de réduire une longue séquence à une série de courtes sous-séquences plus faciles à analyser. Malheureusement, il est difficile d'identifier une série générale de propriétés désirables pour une segmentation. Ainsi, de nombreuses approches ont été proposées, souvent adaptées à une application ou à un type de données particulier. Par exemple, Aggarwal et Cai [AC99] proposent une revue détaillée des différentes techniques de segmentation possibles lorsque les données sont obtenues à partir de séquences vidéo. Dans le cadre de données acquises par capture de mouvements, on distingue aussi différentes approches que nous présentons ci-après.

Utilisation de la vitesse et de l'accélération

Fod *et al.* [FMJ02] soulignent la difficulté bien connue de segmenter un signal temporel, qui constitue un des principaux défis de la technique qu'ils proposent pour la classification des mouvements d'un bras. En particulier, la segmentation qu'ils souhaitent obtenir doit être : consistante, soit produire des segmentations identiques pour différentes répétitions d'un même mouvement ; et complète, soit constituer une partition de l'animation originale. Une des approches étudiées consiste à segmenter le mouvement lorsque les vitesses angulaires de plusieurs joints sont nulles durant un court laps de temps. Un seuil minimal de variation angulaire est utilisé pour éviter la création de multiples segmentations lors d'un mouvement de faible amplitude. Une technique similaire est utilisée par Basu *et al.* [BSC05].

Fod *et al.* [FMJ02] proposent aussi une autre méthode de segmentation basée sur la somme des carrés des vitesses angulaires des degrés de liberté. Avec cette approche, un nouveau segment est introduit lorsque la somme est assez près de zéro, indiquant que la plupart des angles ont une vitesse faible. Le seuil au-dessous duquel un segment est introduit est déterminé empiriquement. Les auteurs comparent cette technique à la précédente et concluent que toutes deux offrent des avantages et des inconvénients. Il est à noter que les squelettes de bras considérés ici ne disposent que de quatre degrés de liberté et qu'il est probable que la segmentation produise des résultats forts différents sur des squelettes plus complexes.

Plutôt que d'utiliser les vitesses angulaires, Bindiganavale [Bin00] segmente l'animation lorsque l'accélération des signaux de mouvements croise zéro. Pour augmenter la robustesse de la segmentation, Zhao [Zha01] combine cette technique à l'évaluation de la courbure de la trajectoire du mouvement et ne procède à la segmentation que si cette courbure atteint un seuil minimal fixé par l'utilisateur. Kim *et al.* [KPS03] appliquent une technique similaire à des mouvements rythmiques. Ils identifient tout d'abord les zéros de l'accélération angulaire à chaque joint. Les temps ainsi identifiés constituent des candidats potentiels pour la segmentation finale. Étant donnée la nature cyclique des animations rythmiques, ces candidats peuvent être utilisés pour extraire une fréquence dominante. La sinusoïde résultante permet ensuite de déterminer précisément la segmentation finale.

Plutôt que d'utiliser les données angulaires ou positionnelles, Kahol *et al.* [KTP04] estiment les forces et les moments appliqués sur les différentes parties du squelette. Les instants où ces forces atteignent un maximum ou un minimum local constituent des points de segmentation potentiels. Certains de ces points sont ensuite rejetés en fonction du résultat d'un classificateur

bayésien, préalablement entraîné sur un petit nombre de séquences.

Utilisation d'un modèle statistique

So et Baciu [SB05b] utilisent un histogramme pour estimer la distribution de probabilité des déplacements subis par des marqueurs placés sur un squelette. La probabilité conjointe de deux déplacements successifs est aussi évaluée de façon similaire. Ces probabilités sont ensuite utilisées pour calculer l'information mutuelle, une métrique entropique permettant d'évaluer le degré d'association entre deux déplacements successifs. Une faible valeur d'information mutuelle indique un changement imprévisible dans le déplacement des marqueurs. Les auteurs proposent donc de segmenter le mouvement lorsque cette mesure entropique est inférieure à un seuil spécifié manuellement.

Évaluation de la dimensionnalité

Barbič *et al.* [BSP+04] souhaitent segmenter une animation en mouvements représentant des activités différentes. Pour ce faire, ils proposent trois techniques susceptibles de détecter les changements importants qui se produisent lorsqu'une animation contient une transition d'une activité à une autre.

La première technique est basée sur l'observation que la dimensionnalité intrinsèque d'un ensemble de poses est plus faible lorsque toutes les poses appartiennent à une même activité. Barbič *et al.* proposent donc d'utiliser la *PCA* sur une fenêtre de poses consécutives de l'animation. Ils évaluent ensuite l'erreur engendrée par une projection sur les dimensions principales ainsi obtenues. Une augmentation brusque de cette erreur est considérée comme l'indication d'un changement d'activité et est utilisée pour déterminer la segmentation.

Dans leur seconde approche, Barbič *et al.* [BSP+04] modélisent les premières poses d'un mouvement à l'aide d'une gaussienne. La matrice de covariance de cette gaussienne est déterminée en appliquant la technique d'analyse probabiliste des composantes principales (*PPCA*). Cette technique permet une modélisation précise de la distribution le long des dimensions principales tandis que les autres dimensions sont modélisées à l'aide d'un bruit gaussien uniforme. Les poses qui sont mal modélisées par la distribution sont considérées comme faisant partie d'une activité différente. Cette technique est aussi utilisée par Liu et McMillan [LM06] pour la compression de mouvements.

Agrégation de poses

La troisième et dernière technique proposée par Barbič *et al.* [BSP+04] modélise l'ensemble des poses de l'animation à l'aide d'un modèle par mélange de gaussiennes [Bis95]. Chaque pose est ensuite associée à une gaussienne par la technique du maximum de vraisemblance. Des poses consécutives appartenant à des gaussiennes différentes sont réputées correspondre à des activités différentes, induisant ainsi une segmentation. En pratique, les auteurs notent que le nombre de gaussiennes du mélange doit être déterminé à l'avance en fonction du nombre d'activités contenues dans les données. De plus, un critère de tolérance doit être utilisé pour éviter les segments trop courts. Une approche similaire est utilisée par Liu *et al.* [LZWM05].

3.3.3 Recherche d'animations

La recherche de séquences dans des données temporelles est un problème général qui a reçu beaucoup d'attention de la part des chercheurs. D'ailleurs, plusieurs algorithmes efficaces ont été développés pour le cas de données présegmentées munies d'une mesure de distance euclidienne. Hetland [Het04] présente d'ailleurs une bonne revue des principaux résultats dans ce domaine.

De tels algorithmes ne sont cependant pas nécessairement adaptés aux données acquises par capture de mouvements. Ceci est dû, d'une part, à la difficulté d'identifier une présegmentation appropriée et, d'autre part, à l'importance d'utiliser une mesure non euclidienne évaluant la distance entre deux séquences après un réalignement. Le reste de cette section présente donc différentes techniques développées dans le cadre précis de la recherche d'animations (*motion query*).

Présegmentation et agrégation

Liu *et al.* [LZWM05] proposent, dans un premier temps, de procéder à l'agrégation (*clustering*) des poses par la technique des k-moyennes (*k-means*) et la *PCA*. Chaque sous-séquence de la banque de données peut ensuite être représentée par une courte signature indiquant sa transition à travers les différentes grappes (*clusters*). Pour effectuer la recherche, une signature est associée au mouvement de la requête et des opérations de comparaison simples permettent d'identifier les signatures similaires dans la banque de données.

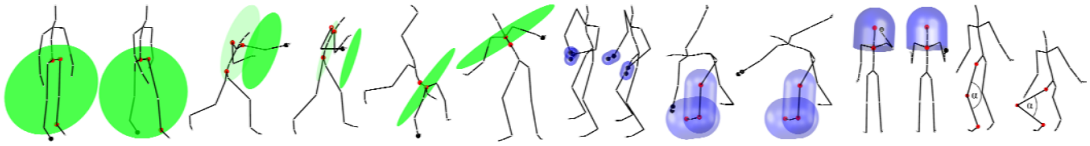


FIG. 3.9 – Quelques-unes des caractéristiques binaires utilisées par Müller *et al.* [MRC05]. De gauche à droite : le pied droit peut être devant ou derrière un plan contenant le pied gauche, la hanche gauche et la racine ; la main gauche peut être éloignée ou rapprochée du corps ; la main droite peut être au-dessus ou en-dessous des épaules ; les mains se touchent ou sont séparées ; la main droite touche le pied gauche ou non ; la main gauche touche la tête ou non ; la jambe gauche est étendue ou pliée. Image tirée de [MRC05].

Basu *et al.* [BSC05] segmentent plutôt les signaux de façon à obtenir des fragments monotones. Ces fragments s'expriment simplement et peuvent être utilisés pour effectuer une recherche efficace.

Utilisation de caractéristiques binaires

Une approche différente consiste à définir un certain nombre de caractéristiques binaires aptes à identifier le comportement général d'une animation. Müller *et al.* [MRC05] utilisent des caractéristiques issues de différentes relations géométriques entre les parties du squelette. Par exemple, les auteurs évaluent si un pied se trouve derrière ou devant le corps, si les jambes sont croisées, si les mains se touchent, etc. Ces caractéristiques, illustrées à la figure 3.9, sont extraites indépendamment pour chacune des poses de la banque de données et de la requête. Ceci permet de simplifier la représentation des mouvements en les encodant sous la forme d'un ensemble de signaux binaires qui accélère la recherche.

Müller et Röder [MR06] étendent ces travaux pour construire, à partir d'un ensemble de mouvements similaires, un patron de mouvement (*motion template*). Celui-ci est construit en alignant temporellement les mouvements et en calculant la moyenne de leurs caractéristiques binaires. Le résultat est une représentation plus "floue" du mouvement indiquant la probabilité de présence d'une caractéristique géométrique à un instant donné. Les patrons de mouvements sont ensuite utilisés pour rechercher une animation de façon plus robuste.

Recherche par poses clés

Liu *et al.* [LZWP03] identifient tout d'abord un certain nombre de poses clés dans la banque de données d'animations. Pour ce faire, ils adaptent la technique d'agrégation non supervisée proposée par Zhuang *et al.* [ZRH98] pour l'extraction d'images clés dans une séquence vidéo. Les poses clés ainsi obtenues constituent une forme de sous-échantillonnage adaptatif qui est appliqué autant à la base de données qu'à l'animation utilisée comme requête. Diverses techniques de recherche peuvent ensuite être appliquées à ces signaux sous-échantillonnés tout en profitant d'un gain d'efficacité important. Les auteurs appliquent un algorithme d'alignement temporel dynamique (*dynamic time-warping*) similaire à ceux qui sont introduits ci-après.

So et Baciu [SB05b] proposent une méthode similaire où les poses clés sont obtenues suite à la segmentation par métrique entropique décrite précédemment.

Alignement temporel dynamique

L'alignement temporel dynamique (*dynamic time-warping* ou *DTW*), traditionellement associée à la recherche en reconnaissance vocale [DP86], est une technique qui consiste à contracter et étirer localement un signal pour le faire correspondre le plus précisément possible à un autre. Bruderlin et Williams [BW95] ont été les premiers à appliquer le *DTW* à des données d'animation. En pratique, cette technique est implantée à l'aide d'un algorithme de programmation dynamique qui identifie la correspondance optimale entre les échantillons des deux signaux.

Plusieurs chercheurs ont proposé d'utiliser le *DTW* pour la recherche de mouvements. Kovar et Gleicher [KG04b] créent tout d'abord un tableau exhaustif des distances inter-poses de la banque de données. Une étape de précalcul basée sur le *DTW* permet ensuite d'identifier toutes les sous-séquences similaires à l'intérieur de ce tableau. Le résultat, nommé réseau de correspondances (*match web*), peut ensuite permettre d'extraire rapidement les mouvements similaires à une requête.

Forbes et Fiume [FF05] développent une approche similaire tout en ajoutant une optimisation importante. Plutôt que de précalculer tout le réseau de correspondances, ils identifient une pose caractéristique dans la requête et extraient toutes les poses similaires de la banque de données. Seul le voisinage des poses candidates ainsi obtenues est utilisé pour la recherche d'un alignement par *DTW*. Les meilleurs alignements constituent les résultats de la recherche.

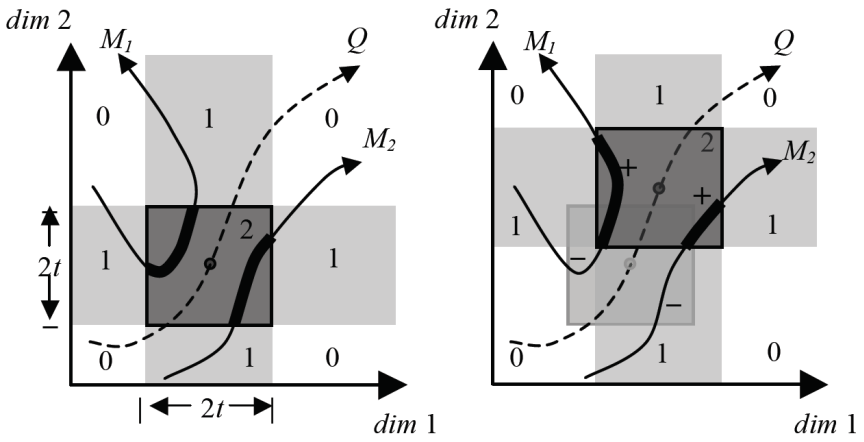


FIG. 3.10 – So et Baciu [SB06] proposent la technique de balayage par hypercube, ici illustrée en 2D. Un hypercube de taille définie par l'usager progresse le long de la requête Q. Une sous-séquence est similaire si au moins une de ses poses tombe à l'intérieur de chacun des hypercubes ainsi générés. Image tirée de [SB06].

Keogh *et al.* [KPZ$^+$04] soulignent que la flexibilité du *DTW* peut permettre l'alignement non souhaitable de signaux d'animation passablement différents. Ils proposent ainsi de ne permettre que des contractions ou des étirements uniformes sur toute la durée d'un signal. L'algorithme proposé compare deux signaux à l'aide de tous les réalignements uniformes possibles et conserve le meilleur. Les auteurs proposent aussi d'accélérer la recherche en calculant une enveloppe minimale pour les signaux de la banque de données.

La technique de balayage par hypercube (*hypercube sweeping*) de So et Baciu [SB06] se rapproche des méthodes de recherche par *DTW* bien que l'alignement optimal ne soit pas directement déterminé. Les auteurs proposent tout d'abord de représenter la requête et la banque de données comme des trajectoires dans l'espace des degrés de liberté. La technique consiste ensuite à faire progresser un hypercube le long de la trajectoire de la requête, tel qu'illustré à la figure 3.10. Une sous-séquence est considérée similaire si elle intersecte chacun des hypercubes ainsi générés. La taille de l'hypercube est spécifiée par l'usager et permet de contrôler la sensibilité de la recherche. Comme les hypercubes sont alignés sur les axes, la comparaison peut être effectuée beaucoup plus efficacement que les techniques évaluant la distance euclidienne.

3.3.4 Modèles statistiques

Un certain nombre de méthodes d'analyse visent à construire un modèle statistique général d'une animation ou d'un mouvement. Un tel modèle peut ensuite être utilisé de multiples façons, par exemple pour la synthèse de mouvements originaux [TH00, LWS02], pour l'étude des variations de style [BH00], pour diriger un algorithme de cinématique inverse [GMHP04] ou pour classifier [PRM00, RPE$^+$05] et reconnaître [WB99] les mouvements.

Ces différentes méthodes se distinguent non seulement par les applications visées mais aussi, et surtout, par le modèle statistique retenu. Nous présentons maintenant ces différents modèles.

Modèle de Markov caché

Une chaîne de Markov (*Markov chain*) est un modèle statistique simple constitué d'un ensemble fini d'états organisés en un graphe dirigé pondéré [Bis95]. La pondération associée à chaque arc indique la probabilité de la transition d'un état à un autre. Un modèle de Markov caché (*Hidden Markov Model* ou *HMM*) étend ce modèle en associant à chaque état une distribution de probabilité permettant de prédire les valeurs les variables observables. Ces distributions sont indépendantes les unes des autres. Formellement, un modèle de Markov caché peut s'exprimer par

$$Pr(s_{t+1} = i | s_t = j) = \Pi(i, j) \quad \text{et}$$
$$\mathbf{y}_t \sim \mathcal{D}(s_t) \ ,$$

où s_t est l'état courant, $\mathcal{D}(s_t)$ la distribution de probabilité associée à cet état et $\mathbf{y}_t$ le vecteur des variables observables. Le problème de modélisation consiste ainsi à inférer le nombre d'états et les paramètres des distributions de probabilité en fonction des données observées. Plusieurs approches sont connues pour résoudre ce problème dans un contexte général [Rab89].

Dans le contexte de données d'animation, Bowden [Bow00] construit un *HMM* en procédant tout d'abord à une agrégation des poses par la méthode des k-moyennes (*k-means clustering*), où le nombre de grappes est déterminé manuellement. Chaque grappe devient ensuite un état du *HMM* correspondant à une distribution de probabilité dont la moyenne et la matrice de covariance peuvent être calculées facilement à partir des données. Il est aussi possible d'estimer la probabilité de transition entre chaque paire de grappes en dénombrant le nombre de poses consécutives se trouvant dans ces deux grappes.

Tanco et Hilton [TH00] construisent un modèle similaire auquel ils ajoutent un second niveau permettant de faire correspondre chaque état de la chaîne de Markov à un segment de

mouvement tiré de la banque de données originale. Ceci permet, entre autres, la synthèse de mouvements originaux.

Brand et Hertzmann [BH00] utilisent un modèle de Markov caché où la distribution de probabilité de chaque état est une gaussienne multidimensionnelle. Chacune de ces gaussiennes est munie d'un certain nombre de paramètres permettant de faire varier localement la moyenne et la matrice de covariance. Ceci vise à offrir un contrôle séparé de la structure et du style d'un mouvement : les paramètres du *HMM* régissent la structure temporelle du mouvement tandis que les paramètres des gaussiennes affectent le style.

Systèmes dynamiques linéaires

Une chaîne de Markov modélise le comportement temporel d'un système par des transitions entre un nombre fini d'états. À l'intérieur d'un état, par contre, le système est modélisé par une distribution de probabilité uniforme. En d'autres mots, un état est vu comme un nuage de points indépendants les uns des autres. On sait cependant qu'un mouvement prend plutôt la forme d'une trajectoire bien structurée de poses. Certains auteurs ont donc proposé un modèle différent où l'évolution locale d'un état n'est plus gouverné par une simple distribution de probabilités, mais plutôt un système dynamique linéaire.[2] Lorsque combiné à une chaîne de Markov, on obtient un *switching linear dynamic system* ou *SLDS*.

Un *SLDS* peut ainsi être vu comme un système à deux niveaux où le premier niveau est une simple chaîne de Markov. Dans le deuxième niveau, plutôt que d'associer une distribution de probabilité à chaque état comme le ferait un *HMM*, on lui associe des paramètres contrôlant un système dynamique linéaire. Formellement, le *SLDS* peut donc se décrire par

$$\begin{aligned} &Pr(s_{t+1} = i | s_t = j) = \Pi(i,j) \qquad \text{et} \\ &\mathbf{x}_{t+1} = \mathbf{A}(s_{t+1})\mathbf{x}_t + v_{t+1}(s_{t+1}) \qquad \text{et} \\ &\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + w_t \ , \end{aligned}$$

où s_t est l'état de la chaîne de Markov et $\mathbf{y}_t$ le vecteur des variables observables. L'état $\mathbf{x}_t$ du processus dynamique linéaire est contrôlé par la matrice $\mathbf{A}$ et le bruit gaussien v_t, tous deux dépendants de l'état actuel de la chaîne de Markov. La matrice $\mathbf{C}$ et un second bruit gaussien w_t permettent de relier l'état du processus dynamique aux variables observées. Pavlović *et al.* [PRM00] montrent trois techniques permettant d'inférer approximativement les paramètres de ce modèle à partir de données vidéo d'animations humaines.

[2] Par système dynamique, on entend ici un système dont l'état évolue en fonction du temps. Il ne faut pas confondre ce concept avec celui, plus commun en infographie, d'un comportement régi par les lois de la mécanique.

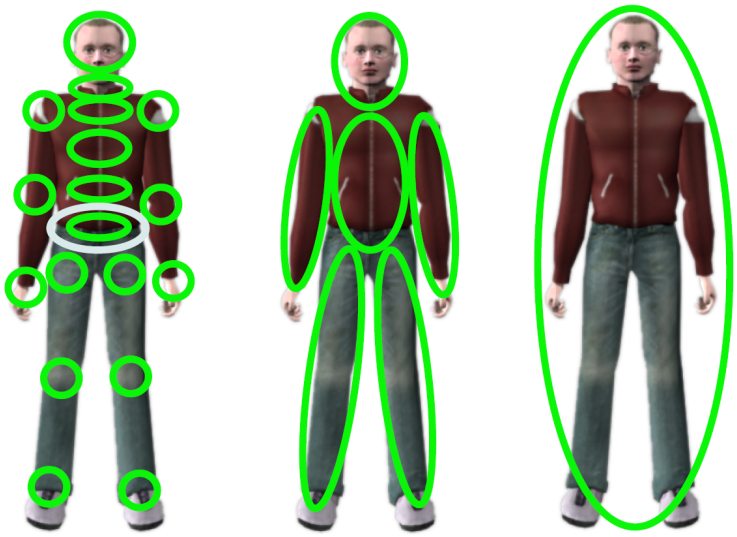


FIG. 3.11 – Hiérarchie proposée par Ren *et al.* [RPE+05] pour la génération de modèles statistiques. À gauche, le niveau le plus bas contient tous les joints du squelette. Au centre, le second niveau contient des ensembles de joints groupés par membre. À droite, tous les joints sont combinés en un seul vecteur. Image tirée de [RPE+05].

Li *et al.* [LWS02] appliquent une technique similaire à des données acquises par capture de mouvements. Contrairement à un *SLDS* classique, cependant, ils ne permettent pas un changement de l'état s_t à chaque instant. En fait, un même système dynamique linéaire est utilisé pour modéliser une séquence d'échantillons assez longue (nommée *motion texton*), suite à quoi un nouveau système dynamique est choisi par une transition de la chaîne de Markov.

Autres approches

Ren *et al.* [RPE+05] évaluent trois techniques différentes pour construire un modèle statistique capable de reconnaître un mouvement naturel. Les techniques utilisées tour à tour sont les mélanges de gaussiennes, les *HMM* et les *SLDS*. Contrairement aux approches présentées précédemment, Ren *et al.* ne modélisent pas simultanément tous les joints du squelette. Ils proposent plutôt de séparer le squelette en ses parties constituantes, organisées de façon hiérarchique tel qu'illustré à la figure 3.11. Les données associées à chacune de ces parties sont ensuite modélisées indépendamment. Pour l'évaluation globale, seule la partie du squelette générant le pire score de classification est retenue, permettant ainsi de détecter des erreurs qui ne se produiraient que sur un seul joint.

Certains chercheurs ont proposé d'utiliser les machines à vecteurs de support (*support vector machines* ou *SVM*) pour classifier les poses. Arikan *et al.* [AFO03] les utilisent pour faciliter l'annotation des sous-séquences contenues dans une banque de données tandis que Ikemoto et Forsyth [IF04] s'en servent pour évaluer le réalisme d'un mouvement. Les *SVM* sont des classificateurs résultant d'un apprentissage supervisé : l'utilisateur indique un certain nombre de poses possèdant une caractéristique et d'autres ne la possédant pas. Le système généralise ensuite ces exemples à l'ensemble des données en construisant une surface de séparation maximale (un hyperplan pour un *SVM* linéaire). Une excellente introduction aux *SVM* est présentée par Burges [Bur98].

Grochow *et al.* [GMHP04] modélisent l'espace des poses à l'aide d'un *scaled gaussian process latent variable model*. Ce type de modèle, introduit par Lawrence [Law04], est une extension des processus gaussiens [Bis95]. Le but, ici, est d'identifier une transformation non-linéaire depuis l'espace des poses vers un espace latent à beaucoup plus faible dimensionnalité. Un mécanisme similaire aux fonctions de base radiales (*radial basis functions* ou *RBF*) permet le passage de l'espace latent vers l'espace des poses de façon à maximiser la vraisemblance de la pose obtenue en fonction des données d'entraînement.

3.3.5 Graphes de mouvements

Une autre technique d'analyse consiste à organiser les données acquises par capture de mouvements sous la forme d'un graphe dirigé. Un tel graphe permet en effet d'illustrer les façons dont les mouvements présents dans les données peuvent être réarrangés. Ceci dans le but de synthétiser une animation composée d'une nouvelle séquence de mouvements. Ce genre de structure est très similaire aux arbres de mouvements (*move trees*), qui sont depuis longtemps l'approche la plus utilisée dans les applications temps réel et les jeux vidéo [Men00, MBC01].

Cette section présente dans un premier temps les graphes de mouvements non paramétrés, où chaque arête correspond à un mouvement unique. Nous distinguons deux types d'approches : celles où un graphe complexe est simplifié pour en extraire une structure efficace et celles où un graphe simple est raffiné jusqu'à l'obtention d'une structure plus flexible. En troisième lieu, nous nous attardons aux graphes paramétriques, où un arc peut encoder une grande gamme de mouvements similaires.

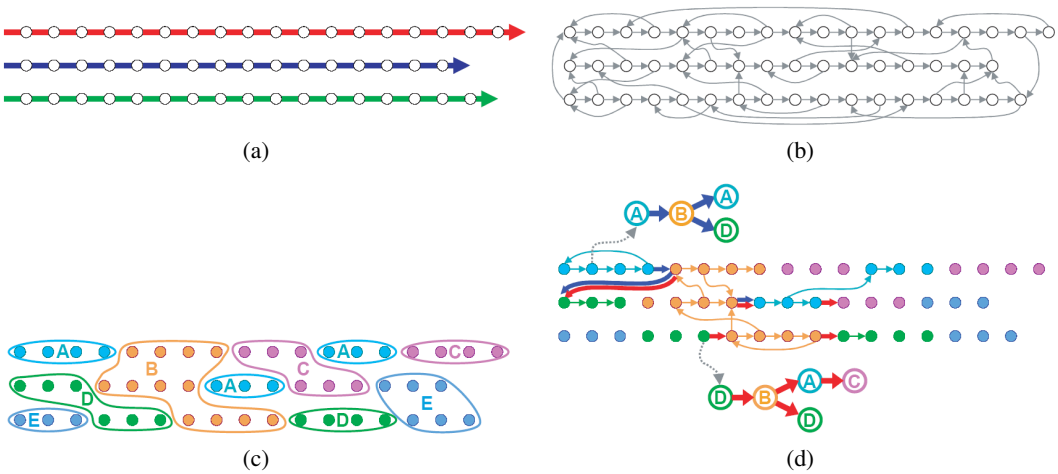


FIG. 3.12 – Construction de graphe de mouvements proposée par Lee *et al.* [LCR+02]. (a) Un graphe reliant chaque pose de la banque de données est créé. (b) Les transitions valides sont identifiées. (c) Les poses similaires sont agrégées. (d) Un arbre de grappes (*cluster tree*) est associé à chaque pose pour lui permettre d'identifier les grappes atteignables, ici deux de ces arbres sont illustrés. Image tirée de [LCR+02].

Simplification d'un graphe complexe

Une première approche pour la création de graphes de mouvements non paramétrés consiste à créer initialement un graphe reliant un très grand nombre de noeuds. Pour beaucoup d'applications, cependant, un tel graphe peut s'avèrer inefficace, voire inutilisable. Il est donc simplifié de différentes façons pour obtenir une représentation à plus haut niveau.

Arikan et Forsyth [AF02] construisent tout d'abord un très grand graphe comprenant un arc entre chaque paire de poses similaires. Ce graphe est ensuite simplifié itérativement en procédant à l'agrégation des arcs par la méthode des k-moyennes. Le résultat est une hiérarchie allant d'un graphe très grossier jusqu'au graphe complet. Les auteurs introduisent ensuite une technique de recherche hiérarchique qui examine ces graphes de façon à identifier un parcours respectant les contraintes spécifiées par l'usager.

Une technique similaire est introduite par Lee *et al.* [LCR+02], qui proposent de créer une très grande chaîne de Markov comprenant un état pour chaque pose de la banque de données. La probabilité associée à chaque transition est déterminée par la distance entre les deux poses. Pour faciliter la recherche d'un mouvement particulier, les auteurs construisent une seconde structure, plus générale, en procédant à l'agrégation des poses de la banque de données. Suite à cette agrégation, chaque pose est associée à un arbre (*cluster tree*) indiquant les grappes qui

peuvent être atteintes efficacement. Ceci permet de déterminer rapidement quels mouvements sont réalisables à partir d'une pose donnée, une opération essentielle pour la synthèse interactive d'animations répondant à des contraintes. Les différentes étapes de la technique sont illustrées à la figure 3.12.

Raffinement d'un graphe simple

Une seconde approche possible pour la création de graphes non paramétrés consiste à débuter avec un graphe très simple, n'encodant que les transitions triviales. Un tel graphe ne permet cependant pas d'identifier de nouvelles transitions. Il doit donc être raffiné de façon à lui ajouter des noeuds et des arcs.

Une approche du genre est proposée par Kovar *et al.* [KGP02]. Ils construisent un premier graphe grossier en plaçant un noeud au début et à la fin de chaque séquence. Chaque paire de noeuds ainsi créée est reliée par un arc, ce qui génère un graphe non connexe très simple. Les auteurs proposent ensuite de connecter les parties de ce graphe en identifiant des paires de poses candidates pour l'insertion d'une transition. Pour déterminer si deux poses peuvent être candidates, les auteurs génèrent un nuage de points 3D autour du squelette des poses en question. Une transition est ajoutée entre deux poses lorsque ces nuages de points sont assez rapprochés. La création d'une transition implique l'ajout de deux noeuds aux poses candidates ainsi que d'un arc les reliant. Le mouvement correspondant à l'arc est obtenu en combinant (*blending*) les sous-séquences avoisinant les poses choisies.

Gleicher *et al.* [GSKJ03] remarquent que le graphe précédent permet de synthétiser un mouvement efficacement lorsqu'on connaît les contraintes à l'avance, mais qu'il s'avère inapproprié dans le contexte d'une application interactive où les contraintes changent constamment. Pour ce genre d'application, il est nécessaire de disposer d'un graphe comprenant des noeuds fortement connectés (*hub nodes*) à partir desquels l'animation peut enchaîner sur un grand nombre de mouvements différents (voir figure 3.13). Pour construire de tels graphes, ils proposent un outil interactif aidant l'utilisateur à identifier les poses présentes à plusieurs reprises dans la banque de données. Ces poses deviennent des noeuds du graphe d'où partent un grand nombre d'arcs. À la différence de Kovar *et al.* [KGP02], les transitions sont effectuées en ajustant le voisinage des poses entrantes et sortantes, à chaque noeud du graphe, de façon à ce que les degrés de liberté et leurs dérivées soient continus peu importe l'arc d'arrivée et l'arc de sortie. Ceci évite d'avoir à synthétiser de nouveaux mouvements de transition.

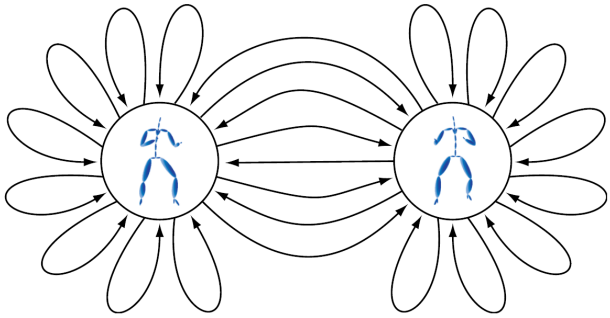


FIG. 3.13 – Gleicher *et al.* [GSKJ03] proposent de créer un graphe contenant des noeuds fortement connectés permettant d'enchaîner rapidement sur un grand nombre de mouvements différents. Chaque noeud représente une pose et chaque arc est un mouvement. Image tirée de [GSKJ03].

Plutôt que de forcer directement la présence de noeuds fortement connectés dans la graphe, d'autres auteurs [LL06b, SMM05] ont plutôt proposé d'extraire un certain nombre de propriétés du graphe original permettant de l'utiliser dans une application interactive. Notons aussi que des structures de graphes ont été utilisées dans le contexte d'applications particulières [AFO03, CLS03, KPS03, LCF05].

Graphes paramétriques

Des structures de graphes où chaque arc représente une gamme de mouvements similaires ont aussi été proposées. Les mouvements contenus dans un arc peuvent être combinés (*blending*) de façon à couvrir un espace continu de possibilités, puis être paramétrés suivant des caractéristiques intuitives. Par exemple, un arc pourrait contenir un grand nombre d'animations d'un coup de poing et être paramétré de façon à déterminer quelle combinaison de mouvements permet de générer un coup atteignant un endroit précis (voir section 3.3.1). Certains auteurs ont proposé des techniques de paramétrisation manuelles adaptées à des applications particulières [PSKS04], tandis que d'autres ont développé des techniques de paramétrisation automatiques [KG04b, KS05].

Étonnamment, la première utilisation d'un graphe paramétrique est antérieure aux approches par graphes de mouvements décrites précédemment. Rose *et al.* [RCB98] proposent en effet une technique qu'ils nomment "verbes et adverbes". Dans celle-ci, chaque verbe correspond à une action particulière et chaque adverbe est un paramètre contrôlant l'allure de

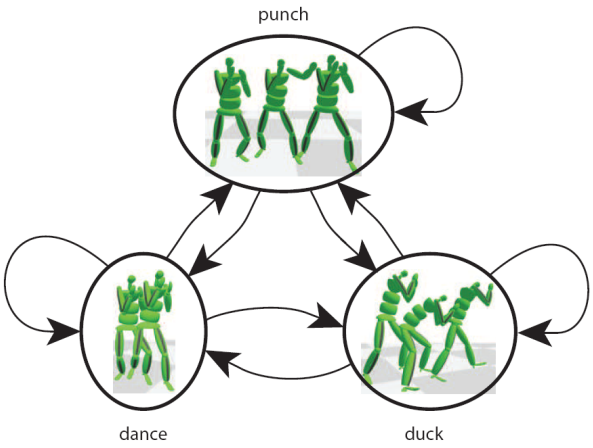


FIG. 3.14 – Graphe paramétrique proposé par Heck et Gleicher [HG07]. Chaque noeud contient une gamme de mouvements similaires qui peuvent être combinés (*blending*). Les arcs représentent des transitions possibles entre deux mouvements. Image tirée de [HG07].

cette action. Contrairement aux techniques précédentes, cependant, Rose *et al.* assument que les données sont préalablement segmentées et que le graphe des verbes est construit manuellement.

Plus récemment, certains chercheurs se sont penchés sur les façons d'intégrer les principes des "verbes et adverbes" aux techniques de construction automatique de graphes présentées précédemment. Shin et Oh [SO06] construisent tout d'abord un graphe à l'aide de la technique de Gleicher *et al.* [GSKJ03] qui leur permet d'obtenir des noeuds fortement connectés. Tous les arcs connectant une même paire de noeuds sont ensuite explorés et ceux correspondant à des mouvements similaires sont regroupés. Les groupes de mouvements ainsi créés sont paramétrés suivant la technique de Kovar et Gleicher [KG04b], et de nouveaux mouvements répondant à des contraintes particulières peuvent ensuite être synthétisés facilement.

Contrairement aux structures précédentes, les noeuds du graphe proposé par Heck et Gleicher [HG07] ne représentent pas une pose unique mais bien un ensemble de mouvements, tel qu'illustré à la figure 3.14. Ces noeuds sont créés semi-automatiquement. Pour un noeud donné, l'usager sélectionne une animation qui est recherchée dans la banque de données à l'aide de la technique de Kovar et Gleicher [KG04b]. Le noeud est ensuite associé à l'ensemble des mouvements similaires ainsi obtenus puis paramétré selon certaines caractéristiques intuitives. L'usager peut ensuite créer des arcs entre ces noeuds de façon à indiquer la possibilité d'une transition d'un groupe de mouvements à un autre. Pour chaque arc ainsi ajouté, le système détermine par

échantillonnage une correspondance entre les paramètres du noeud de départ et ceux du noeud d'arrivée. Ceci permet d'éviter les transitions aberrantes qui utiliseraient des paramètres incompatibles.

Safonova et Hodgins [SH07] créent tout d'abord un graphe de mouvements simple en utilisant la technique de Lee *et al.* [LCR+02]. Ils génèrent ensuite un graphe de mouvements interpolés dans lequel chaque noeud est une paire d'états du graphe initial auquel est ajouté un facteur de pondération. Ce nouveau graphe contient en fait toutes les animations qui peuvent être produites en effectuant deux parcours distincts dans le graphe de mouvements initial et en combinant (*blending*) les résultats. Ce graphe est ensuite compressé et traité de façon à permettre la recherche efficace d'une animation répondant à des contraintes définies par l'utilisateur.

3.4 Compression d'animations squelettiques

Un certain nombre de travaux de recherche récents sont reliés directement ou indirectement à la compression d'animations squelettiques. Nous présentons maintenant ces différents travaux.

Dans la première partie de cette section nous nous attardons à différents travaux s'intéressant au problème de l'évaluation de la fidélité d'une animation. Par la suite nous présentons un certain nombre de techniques exploitant une représentation simplifiée d'une animation mais dont l'objectif n'est pas de compresser les données. Finalement, nous revisons en détail les travaux s'intéressant directement au stockage de banques de données d'animations compressées.

3.4.1 Évaluation de la fidélité

Un aspect important des techniques de compression qui nous intéressent consiste à évaluer la fidélité des animations reconstruites. En effet, une compression avec pertes engendre nécessairement des erreurs et il est important de pouvoir estimer l'impact visuel de celles-ci. Cette évaluation est cependant complexe et dépend d'une foule de paramètres dont certains varient en fonction de la sensibilité de l'observateur.

En psychologie, la perception humaine du mouvement a été étudiée entre autres par Johansson [Joh73]. Ses expériences utilisant la projection de points lumineux ont montré qu'un observateur humain pouvait reconnaître presqu'instantanément un mouvement biologique, même lorsqu'il ne dispose que d'un petit nombre de points de référence.

Si ces travaux aident à mieux comprendre les mécanismes de la perception humaine du mouvement, ils ne proposent cependant pas de méthode permettant d'évaluer la fidélité d'une animation. En infographie, certains chercheurs se sont penchés sur le problème de l'évaluation de la fidélité d'images statiques [Dal93, TH94, WFM01] ou d'images animées [van04, Lub97, WS98]. Les animations 3D qui nous intéressent sont cependant indépendantes de la position de la caméra, ce qui rend impossible l'utilisation des techniques précédentes. À notre connaissance, aucune métrique n'a été proposée pour évaluer la fidélité d'une animation 3D de personnage.

Certains chercheurs ont néanmoins tenté de déterminer des critères permettant d'évaluer le réalisme ou la qualité d'une animation 3D d'objets simples. Des modèles ont été proposés pour évaluer la sensibilité humaine aux erreurs d'accélération [HB00] ou aux perturbations lors de collisions [ORC99, OD01]. O'Sullivan *et al.* [ODGK03] proposent une métrique permettant de quantifier la fidélité visuelle d'une animation résultant d'une simulation physique. Harrison *et al.* [HRv03], quant à eux, évaluent la sensibilité du système visuel humain à des changements graduels ou abrupts de la vitesse d'un objet.

Dans le cas de personnages anthropomorphiques, Hodgins *et al.* [HOT98] montrent que la perception des caractéristiques du mouvement dépend du modèle géométrique utilisé pour représenter le personnage animé. Oesker *et al.* [OHJ00] étudient l'importance du niveau de détail d'une animation et concluent qu'un observateur peut distinguer de façon inconsciente des variations présentes dans une animation, même s'il n'est pas en mesure d'en indiquer précisément la nature. Pollick *et al.* [PHM03] proposent une expérience visant à identifier les caractéristiques qui permettent à un observateur de distinguer deux mouvements différents. Wang et Bodenheimer [WB03, WB04] ont déterminé empiriquement la durée d'une transition entre deux animations d'un personnage de façon à minimiser les artefacts perçus. Reitsma et Pollard [RP03] proposent une métrique perceptuelle pour mesurer les erreurs dans le mouvement ballistique d'un personnage animé. Ren *et al.* [RPE$^+$05] proposent d'analyser des données acquises par capture de mouvements pour développer une métrique permettant de quantifier le réalisme d'une animation. Harrison *et al.* [HRv04] examinent l'effet d'un changement de la longueur des membres d'un personnage animé et concluent qu'un changement allant jusqu'à 20% de la longueur d'un membre est généralement indétectable lorsque l'attention de l'observateur n'est pas focalisée sur ce membre.

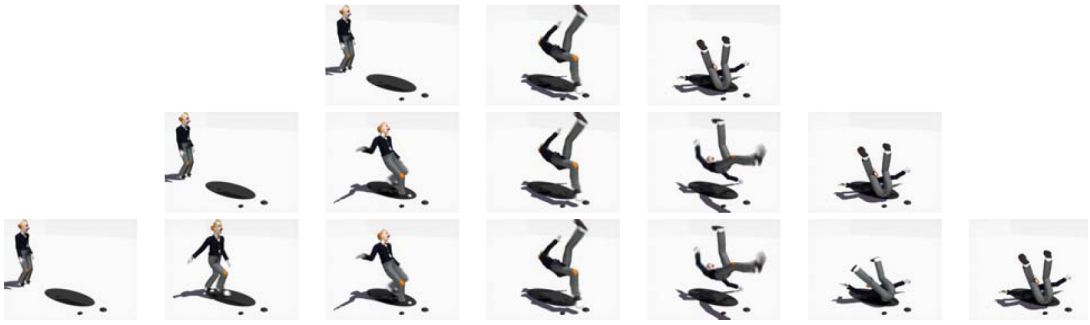


FIG. 3.15 – Technique de simplification temporelle de Assa *et al.* [ACCO05]. De haut en bas : la séquence simplifiée à uniquement 3 poses, à 5 poses, puis à 7 poses. Image tirée de [ACCO05].

3.4.2 Simplification d'animations

Plusieurs chercheurs ont proposé des techniques permettant de simplifier la représentation d'animations avec des objectifs autres que la compression. Ces simplifications sont essentiellement de deux types : réduction du nombre de degrés de liberté (simplification spatiale) et réduction du nombre d'échantillons (simplification temporelle).

Simplification spatiale

La réduction du nombre de degrés de liberté s'effectue en projetant chaque vecteur de poses dans un sous-espace comptant moins de dimensions. La technique de loin la plus utilisée pour déterminer cette projection est la *PCA* (voir section 3.1.2). Celle-ci a été proposée pour la recherche d'animations [FF05], pour le développement de divers outils d'édition [SHP04, BSP$^+$04], et pour la synthèse d'animations [PB02, GBT04, CH05].

Différentes projections non-linéaires ont aussi été proposées. Grochow *et al.* [GMHP04] utilisent une extension des processus gaussiens pour développer une technique de cinématique inverse (voir section 3.3.4). Assa *et al.* [ACCO05] projettent les degrés de liberté à l'aide du *replicated multidimensional scaling* ou *RMDS* [McG68] avant d'effectuer une simplification temporelle. Seward et Bodenheimer [SB05a] utilisent quant à eux des projections à l'aide d'*isomaps* spatio-temporels (*ST-isomap*) [TdL00, JM03].

Simplification temporelle

La simplification temporelle la plus simple est sans conteste le sous-échantillonnage. Celui-ci consiste à conserver un sous-ensemble des échantillons régulièrement espacés. Bon nombre

de techniques utilisant des données acquises par capture de mouvements procèdent tout d'abord à un sous-échantillonnage. Ceci est dû au fait que la fréquence d'acquisition des données est souvent bien supérieure à la fréquence requise pour une utilisation particulière.

D'autres formes de simplifications temporelles plus complexes ont aussi été introduites. Différents auteurs [LT01, KM04] proposent d'identifier les poses clés d'une animation à l'aide d'une technique de simplification de courbes. Assa *et al.* [ACCO05] projettent les données par *RMDS* pour obtenir une courbe de faible dimensionnalité. Cette courbe est ensuite filtrée, puis seules les poses les plus éloignées de cette version filtrée sont conservées de façon à construire un synopsis temporel de l'animation. Un exemple de cette technique est illustré à la figure 3.15.

Liu *et al.* [LGC94] utilisent une représentation en ondelettes des signaux d'animation d'un bras robot à 6 degrés de liberté. Cette représentation simplifiée leur permet d'adopter une approche hiérarchique pour résoudre des problèmes complexes d'optimisation spatio-temporelle [WK88, Coh92]. Cette technique a été réutilisée par Rose *et al.* [RGBC96].

Lee et Shin [LS99] proposent de représenter les signaux d'animation à l'aide d'une hiérarchie de B-splines [LWS97] de façon à en faciliter la manipulation. Les auteurs proposent aussi une représentation plus générale [LS00] utilisant différentes hiérarchies de signaux simplifiés.

3.4.3 Utilisation de compressions génériques

Un certain nombre de travaux se sont penchés sur l'efficacité de techniques génériques de compression d'un signal temporel lorsqu'appliquées à des données d'animation.

Ahmed *et al.* [AHM02] investiguent l'efficacité d'une compression par ondelettes et concluent en l'importance de traiter différemment certains joints, comme les effecteurs terminaux (*end effectors*), qui requièrent une précision plus grande de façon à ne pas introduire d'erreur visuelle dans le mouvement. Li *et al.* [LOT04] utilisent une compression en ondelettes, mais permettent à l'utilisateur de contraindre certaines caractéristiques. Dans une seconde publication, Li *et al.* [LOT05] examinent l'efficacité d'une compression basée sur un prédicteur linéaire. La cinématique inverse est utilisée pour contrôler plus précisément la position des effecteurs terminaux. Etou *et al.* [EON04] étudient quant à eux les propriétés de compression d'une simplification temporelle par sélection de poses.

Le standard MPEG-4 [14491a, 14491b] prévoit la possibilité de transmettre en temps réel et de manipuler des personnages humains virtuels [BSJ04]. Ceci a donné lieu à l'élaboration

d'un standard d'encodage permettant de décrire et animer un squelette [CT99]. En particulier, le contrôle du squelette s'effectue à travers un certain nombre de paramètres bien définis nommés *body animation parameters* ou *BAP*. Capin *et al.* [CPO00] ont étudié le comportement de techniques de compression sans pertes appliquées à ce standard. Le taux de compression peut éventuellement être augmenté en introduisant des pertes par sous-échantillonnage, une technique nommée *frame dropping* dans le standard MPEG-4. Chattopadhyay *et al.* [CBL05] montrent comment le *frame dropping* peut être amélioré en choisissant judicieusement les échantillons à conserver. Chattopadhyay *et al.* [CBL07] proposent aussi le *BAP indexing*, une technique adaptée aux plateformes à faible puissance.

3.4.4 Techniques de compression spécifiques

Dans un poster et un court rapport technique, Lee et Lasenby [LL06a] proposent une technique de compression en ondelettes conçue pour les animations squelettiques. Cette technique se rapproche de celle que nous décrivons au chapitre 4, à la différence que les auteurs utilisent une métrique de qualité basée sur des pondérations indépendantes de l'animation. Ceci résulte en une approximation plus grossière de l'erreur mais leur permet d'utiliser une méthode d'optimisation plus rapide que celle que nous proposons.

Liu *et al.* [LZWM05, LZWM06, LM06] élaborent deux techniques de compression spécifiquement adaptées au stockage d'une banque de données acquises par capture de mouvements. Les approches proposées compressent les données brutes disponibles à la sortie d'un système optique de capture de mouvements, soit une série de points 3D indiquant l'évolution de la position de marqueurs placés sur l'acteur (voir figure 2.2). Notons que ces données ne profitent pas des cohérences inhérentes à la nature hiérarchique du squelettes. De plus, elles doivent être traitées par un système de cinématique inverse global avant de pouvoir être utilisées efficacement dans un système d'animation.

Dans [LZWM05, LZWM06], les auteurs proposent de générer des espaces linéaires de faible dimensionnalité dans lesquels sera projetée chaque pose. Ces espaces sont générés à l'aide d'une stratégie d'agrégation hiérarchique : les poses sont initialement séparées en deux groupes par la technique des k-moyennes, puis une *PCA* est appliquée indépendamment à chaque groupe. Si une projection requiert la conservation d'un trop grand nombre de vecteurs de base pour atteindre un seuil d'erreur spécifique, alors la grappe correspondante est séparée en deux en appliquant à nouveau les k-moyennes. Deux nouvelles bases *PCA* sont ensuite iden-

tifiées. Au final, le stockage de la banque de données requiert l'encodage de chacune des bases (typiquement, 8000 matrices de 62×4) ainsi qu'un vecteur de faible dimensionnalité pour chaque échantillon (typiquement, 2.5 millions de vecteurs de 4 dimensions).

Liu et McMillan [LM06] étendent ces travaux de façon à prendre en considération les cohérences temporelles présentes dans les données. Ils procèdent tout d'abord à une segmentation de l'animation en utilisant l'approche par *PPCA* de Barbič *et al.* [BSP+04] (section 3.3.2). Chacun des segments est ensuite projeté sur un espace linéaire de faible dimension en utilisant la *PCA*. Finalement, une compression temporelle est effectuée en sélectionnant un sous-ensemble des poses et en approximant le reste de l'animation à l'aide d'une interpolation cubique. Pour ce faire, un algorithme vorace sélectionne et ajoute des poses jusqu'à ce que l'interpolation cubique corresponde assez précisément aux données.

La technique de compression de banque de données proposée par Arikan [Ari06] représente chaque degré de liberté à l'aide de trois marqueurs virtuels 3D : un marqueur situé à la position du joint, un dans sa direction principale et un dans une direction perpendiculaire (voir figure 3.16). Arikan justifie le choix d'une telle représentation, similaire à une paramétrisation matricielle [Gra98], en indiquant qu'elle produit un espace plus linéaire[3] et moins sensible à la nature hiérarchique des données. Cette paramétrisation multiplie cependant par trois la taille des données initiales car elle nécessite de stocker neuf scalaires alors qu'une rotation dans l'espace peut être paramétrée avec trois scalaires. De plus, il est possible que, après compression, les trois marqueurs virtuels divergent légèrement et ne représentent plus une rotation. Arikan indique qu'il est alors nécessaire d'identifier la transformation rigide qui approxime le mieux les marqueurs au sens des moindres carrés. Malgré ces inconvénients, Arikan note que la transformation vers cet espace permet d'atteindre de meilleurs taux de compression.

Suite à cette transformation, Arikan divise uniformément toute la banque de données en segments temporels comptant un nombre constant d'échantillons, soit typiquement de 16 à 32 échantillons par segment (130 à 270 ms). À l'intérieur d'un segment, la trajectoire de chaque point 3D est modélisée à l'aide d'une courbe de Bézier cubique déterminée par moindres carrés. La trajectoire locale d'un marqueur est donc un point dans un espace à 12 dimensions : trois dimensions pour chacun des quatre points de contrôle de la courbe de Bézier. L'animation complète devient alors une séquence de points en $12 \times k_{\mathbf{x}}$ dimensions, où $k_{\mathbf{x}}$ est le nombre de joints du squelette.

[3] Arikan fait remarquer qu'avec la plupart des paramétrisations angulaires, une rotation à mi-chemin entre a et b n'est pas nécessairement $(a + b)/2$.

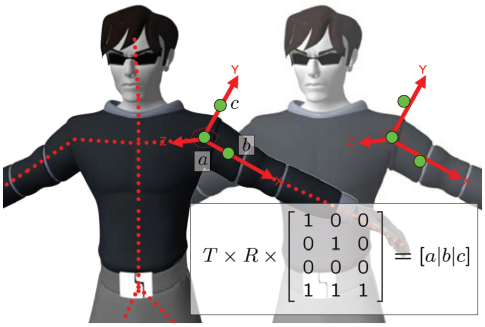


FIG. 3.16 – Paramétrisation des rotations proposée par Arikan [Ari06] pour la compression. À gauche, chaque joint est représenté par trois points 3D. À droite, après une compression avec pertes, ces points ne forment plus nécessairement des vecteurs perpendiculaires. La base orthogonale la plus proche est alors identifiée. Image tirée de [Ari06].

Un algorithme d'agrégation spectrale avec approximation de Nystrom [SM00, FBCM04] est ensuite appliqué à la séquence de points. Pour chaque grappe ainsi formée, une base linéaire de faible dimensionnalité est identifiée par *PCA*. Le stockage compressé d'un segment nécessite de conserver un indice de grappe ainsi que les coefficients du point représentant le segment dans l'espace associé. Les coefficients sont quantifiés à 16 bits et sont encodés par la méthode de Huffman [Huf51].

Arikan souligne l'importance de compresser précisément la position des pieds. Pour ce faire, il encode indépendamment la position des marqueurs virtuels associés aux pieds. Ceux-ci sont transformés par *DCT* et les coefficients résultants sont quantifiés puis encodés par la méthode de Huffman. Ceci permet de contrôler indépendamment la qualité de reconstruction des pieds.

Les travaux antérieurs présentés dans ce chapitre constituent tantôt des bases à partir desquelles nous développons nos recherches, tantôt des outils dont nous faisons usage. Par exemple, le problème de la compression de maillages animés présente plusieurs parallèles avec la compression d'animations acquises par capture de mouvements. On y retrouve entre autre deux types de cohérences : la correlation spatiale entre les sommets du maillage (similaire à la correlation entre les degrés de liberté), et la cohérence temporelle résultant du déplacement des sommets (similaire à la cohérence temporelle des animations acquises par capture de mouvements). Dans les chapitres suivants, nous revenons en détails sur les travaux antérieurs directement reliés aux nôtres en soulignant les principales originalités de nos contributions.

Chapitre 4

Compression en ondelettes

Mr Norrell desires me to inform you that the spells to prevent the flooding of the rivers in the County of Suffolk are now complete.

John Childermass, in a letter to Lord Sidmouth

Ce chapitre s'attarde à l'exploitation des cohérences temporelles à petite échelle en présentant une technique de compression basée sur une représentation en ondelettes adaptée à la nature des données acquises par capture de mouvements. Les travaux présentés dans ce chapitre ont fait l'objet d'une publication à la conférence *Graphics Interface 2007* [BPv07b] et d'un poster au *Symposium on Computer Animation 2007* [BPv07a]. [1]

4.1 Travaux antérieurs reliés

Les travaux antérieurs touchant au domaine de la compression d'animation ont été couverts en détail à la section 3.4. Nous résumons ici les principaux résultats dans ce domaine tout en soulignant nos contributions.

Plusieurs chercheurs ont proposé de simplifier une animation en réduisant la complexité liée au grand nombre de degrés de liberté. À cette fin, la *PCA* a fréquemment été utilisée avec succès [PB02, FF05, SHP04, BSP$^+$04, GBT04, CH05], de même que certaines techniques de projection non-linéaire [GMHP04, ACCO05, SB05a]. Bien qu'il soit envisageable d'utiliser ce type de simplification spatiale dans le cadre de la compression, aucun des auteurs précédents ne s'est spécifiquement intéressé à ce problème.

[1] Dans ce chapitre, n, k, r et $\mathbf{d}$ désignent respectivement $n^{\mathtt{D}}$, $k^{\mathtt{D}}$, $r^{\mathtt{W}}$ et $\mathbf{d}^{\mathtt{D}}$.

Des travaux sur la simplification temporelle des animations ont donné lieu à des techniques d'identification de poses clés [LT01, KM04, ACCO05]. L'objectif, ici, n'est pas d'obtenir une version compressée de l'animation qui soit semblable à l'originale, mais plutôt un ensemble réduit de poses clés représentatives. Les ondelettes et des représentations hiérarchiques semblables ont aussi été utilisées [LGC94, RGBC96, LWS97, LS99, LS00] pour différentes applications autres que la compression.

La simplification temporelle a aussi été utilisée dans le but de compresser des données d'animation. À cette fin, le sous-échantillonnage [CPO00, CBL05], la simplification de poses [EON04], les prédicteurs linéaires [LOT05] et les représentations en ondelettes génériques [AHM02, LOT04] ont été explorées. Les travaux présentés dans ce chapitre appartiennent à cette dernière catégorie. Tout comme Lee et Lasenby [LL06a], nous nous intéressons aux manières d'adapter les compressions en ondelettes génériques de façon à exploiter les particularités des animations squelettiques. À la différence de ces derniers, cependant, nous utilisons une métrique d'erreur qui ne dépend pas uniquement du squelette, mais de toute l'animation. Nous sommes ainsi en mesure de tirer avantage des configurations de poses particulières qui pourraient apparaître au cours d'une animation.

Différentes approches exploitant simultanément les cohérences spatiales et temporelles ont été proposées. Certaines de ces techniques visent à stocker l'évolution de la position des marqueurs acquis par capture de mouvements [LZWM05, LZWM06, LM06]. Lorsqu'on désire connaître les angles associés aux degrés de liberté, comme c'est le cas par exemple pour un jeu vidéo [Lav04], ces techniques exigent de résoudre un système de cinématique inverse global potentiellement coûteux. La compression que nous proposons traite directement les degrés de liberté et échappe donc à cette contrainte.

La technique proposée par Arikan [Ari06] exploite elle aussi les cohérences spatiales et temporelles. La représentation utilisée, une série de marqueurs virtuels pour chaque joint, est moins directe que celle que nous proposons, mais elle permet néanmoins de reconstruire assez rapidement la valeur de chaque degré de liberté. La différence principale avec notre approche vient du fait qu'Arikan analyse les cohérences temporelles sur un très petit nombre d'échantillons (typiquement de 16 à 32) alors que nous les exploitons sur des séquences pouvant compter jusqu'à 5000 poses.

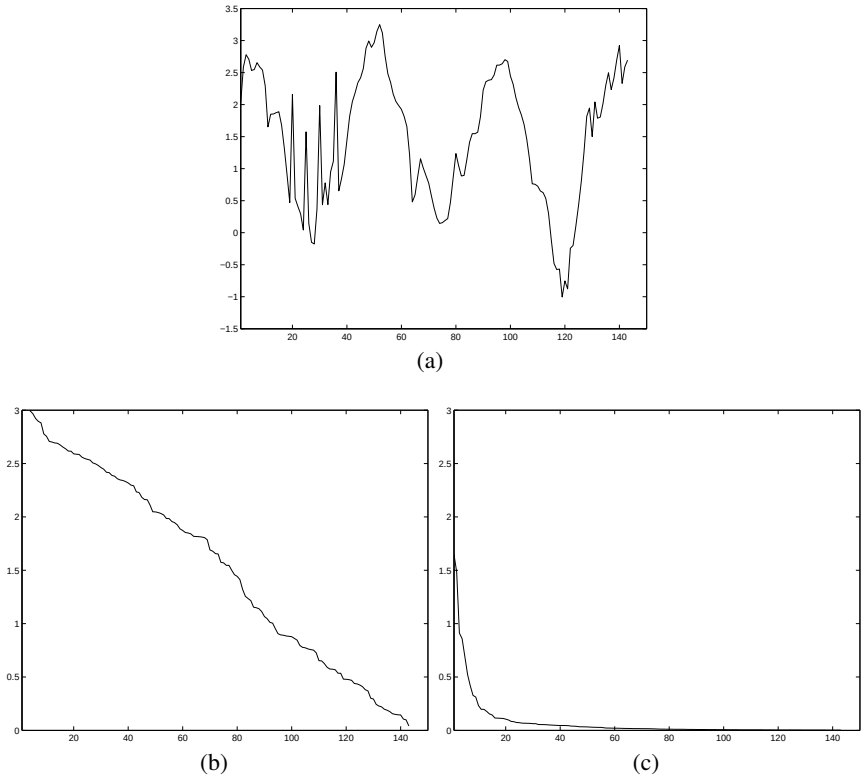


FIG. 4.1 – (a) Un signal typiquement rencontré dans des données de capture de mouvements. Ce signal particulier correspond à la rotation autour de l'axe $\mathbf{x}$ du joint *upperback* dans l'animation 09_05.amc tirée de la *CMU Motion Capture Database* [CMU]. (b) Les valeurs absolues ordonnées des échantillons de ce signal. (c) Les valeurs absolues ordonnées des coefficients résultant d'une transformée en ondelettes. On voit facilement que la transformée en ondelettes produit peu de coefficients de grande valeur, ce qui témoigne de sa capacité à concentrer l'énergie.

4.2 Introduction à la compression en ondelettes

Il est bien connu que la représentation dans une base d'ondelettes d'un signal présentant des cohérences temporelles à petite échelle permet généralement d'en concentrer l'énergie. C'est-à-dire que, dans cette représentation, la plupart des coefficients prendront des valeurs très près de 0. Pour illustrer ceci, la figure 4.1 montre un signal typique, la liste ordonnée des échantillons, ainsi que la liste ordonnée des coefficients résultant d'une transformée en ondelettes. On y voit immédiatement que la distribution des valeurs des coefficients est plus débalancée suite à une transformée en ondelettes, ce qui témoigne de la concentration de l'énergie dans un petit nombre de coefficients.

Ce débalancement des coefficients est la clé des divers algorithmes de compression en ondelettes standards. Ceux-ci exploitent en effet le fait que les coefficients de faibles valeurs ont une influence relativement mineure sur le signal reconstruit.

Ces algorithmes comportent généralement une étape de quantification uniforme (*uniform quantization*) qui consiste à spécifier un certain nombre de casiers (*bins*) uniformément distribués grâce auxquels les coefficients pourront être discrétisés. La taille des données produites par cette quantification peut ensuite être réduite en utilisant une combinaison d'encodages entropiques et *RLE* (*run-length encoding*). Le résultat de cet encodage peut se visualiser comme une réduction du nombre de bits alloués aux coefficients plus faibles au profit des coefficients plus importants.

Cette technique permet d'obtenir de très bons résultats mais n'offre pas un contrôle précis sur la taille finale des données compressées. En effet, le nombre de casiers utilisés pour la quantification a une influence sur l'entropie qui, à son tour, influence l'efficacité des encodages entropiques et *RLE*. Cette relation est cependant complexe et fait en sorte qu'il est difficile d'évaluer les modifications à apporter à la quantification pour obtenir un changement précis de la taille finale. Comme nous le verrons plus tard, ce genre de contrôle est essentiel pour la mise au point d'une compression en ondelettes adaptée aux données acquises par capture de mouvements.

Pour pallier à ce problème, nous ajoutons une étape supplémentaire préalable à la quantification : la troncation. Il s'agit en fait d'annuler un certain nombre de coefficients et ce, même si la granularité de la quantification serait assez fine pour les exprimer. La troncation offre donc un contrôle direct sur le nombre de coefficients d'ondelettes non nuls utilisés pour représenter le signal. En outre, le nombre de ces coefficients est fortement corrélé avec la taille des données compressées. Cette étape supplémentaire permet donc un contrôle précis de la taille des données compressées sans pour autant réduire la précision des coefficients importants.

Le processus complet de compression en ondelettes, illustré à la figure 4.2(a), consiste à enchaîner ces différentes étapes. La compression qui nous occupe utilise une quantification fixe à fine granularité. Ainsi le processus simplifié illustré à la figure 4.2(b) nous permet d'estimer le taux de compression et le signal reconstruit en fonction du nombre de coefficients conservés lors de la troncation. Nous introduisons d'ailleurs la notation $\mathrm{trunc}(s, m)$ pour dénoter le signal obtenu suite à la transformation en ondelettes du signal s, suivi d'une troncation conservant les m plus grands coefficients, suivi de la transformation en ondelettes inverse.

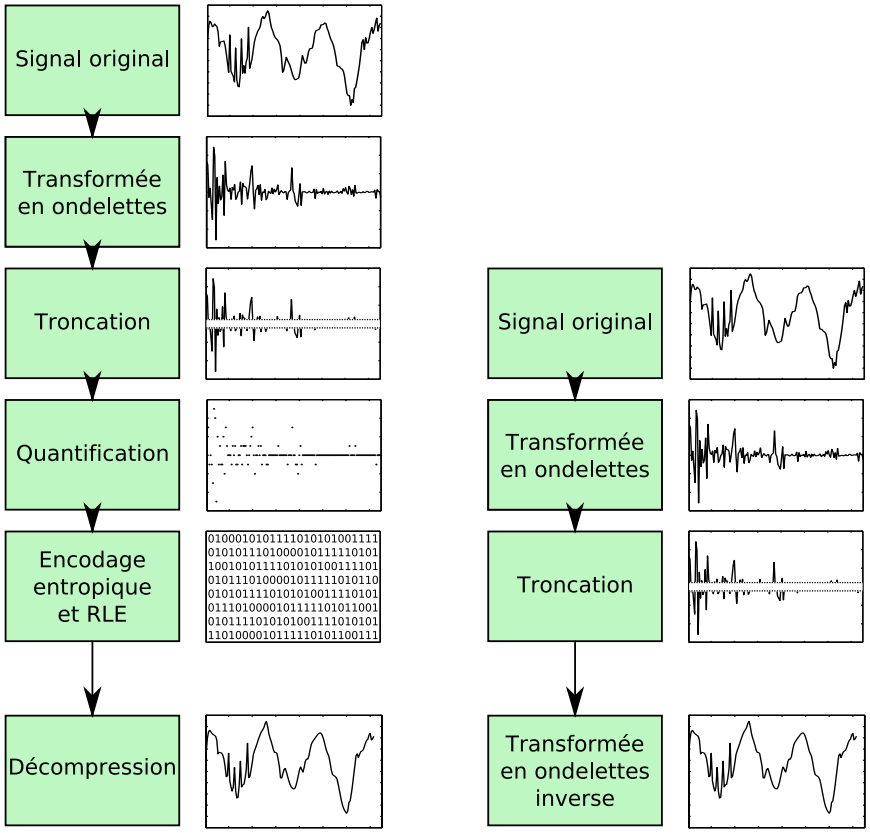


FIG. 4.2 – (a) Les différentes étapes de la compression en ondelettes. (b) La version simplifiée correspondant à $\mathrm{trunc}(s, m)$ lorsque le signal original est s et que la troncation conserve m coefficients non nuls.

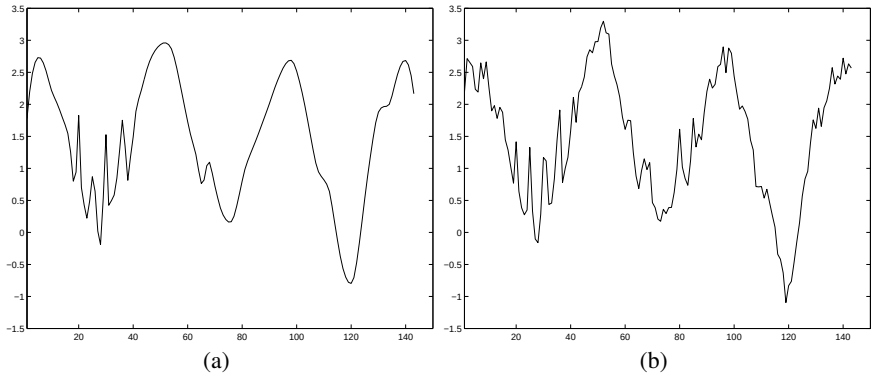


FIG. 4.3 – (a) Le signal de la figure 4.1, reconstruit après avoir subi une compression en ondelettes agressive. (b) Ce même signal reconstruit après une compression *DCT*. Les oscillations visibles dans (b) et absentes du signal original produisent des vibrations perceptibles dans l'animation.

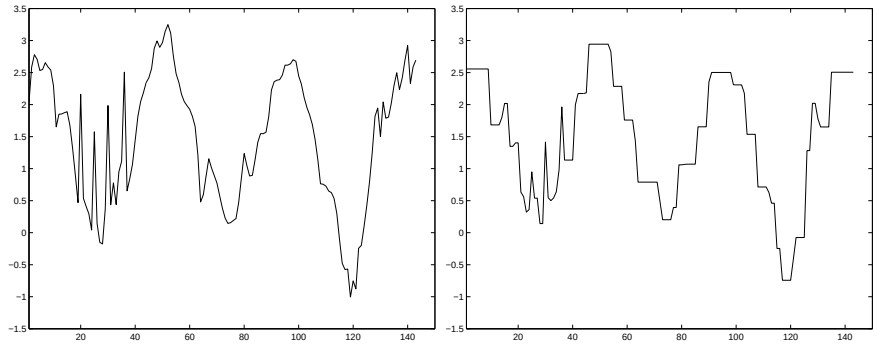
En terminant, il est intéressant de noter que la transformation en ondelettes n'est pas la seule à concentrer l'énergie d'un signal présentant des cohérences temporelles à petite échelle. En fait, la figure 4.2 peut être modifiée pour utiliser n'importe quelle transformation inversible, donnant naissance au codage par transformation général (*transform coding*). Par exemple, une autre transformation fréquemment utilisée pour des signaux temporels est la transformée en cosinus discrète (*discrete cosine transform* ou *DCT*). Les fonctions de base de la *DCT* ont cependant un support infini et, par conséquent, la troncation peut avoir des effets non locaux. La figure 4.3 compare les résultats de la troncation lors d'une compression en ondelettes et d'une compression par *DCT*. En pratique, lorsqu'on l'applique à des données acquises par capture de mouvements, les oscillations dues à la *DCT* produisent des vibrations clairement visibles dans une animation. C'est pour cette raison que nous préférons utiliser la transformée en ondelettes.

4.3 Définition d'une base d'ondelettes

L'introduction qui précède omet une facette cruciale de ce type de compression : la définition de la base d'ondelettes choisie. En effet, il existe de nombreuses bases fonctionnelles possédant les propriétés des ondelettes et elles ne sont pas toutes adaptées à l'encodage des signaux temporels qui composent les données d'animation qui nous intéressent.

Des bases d'ondelettes sont fréquemment utilisées pour la compression d'images [15444] ou de séquences vidéos [KP97], mais ce genre de données diffère largement des signaux temporels typiques. Les signaux audio, quant à eux, partagent de nombreuses similarités avec les signaux issus de la capture de mouvements. Cependant, les standards les plus utilisés (MPEG-1 layer III ou MP3 [11172], MPEG-2 *Advanced Audio Coding* ou AAC [13818]) ne sont pas basés sur les ondelettes. Certains autres formats se fondent plutôt sur la transformée en ondelettes par paquets (*wavelet packet transform*) [Mal98], mal adaptée au contrôle précis que nous souhaitons avoir sur la taille compressée d'un signal.

L'approche que nous avons retenue, plus rarement utilisée en encodage audio, se base sur le schéma de *lifting* (*lifting scheme*) pour la construction d'une base d'ondelettes. Nous avons retenu cette approche pour l'efficacité des algorithmes qui en résultent ainsi que pour la possibilité de construire une base adaptée à nos besoins. En effet, le schéma de *lifting* est très flexible et permet de construire des ondelettes dites de *seconde génération* : ondelettes définies sur un intervalle fini, sur des échantillons irréguliers ou munies d'un produit scalaire quelconque.



(a) (b)

FIG. 4.4 – (a) Le signal original. (b) Ce même signal reconstruit après une compression utilisant la base d'ondelettes de Haar. Les discontinuités dans le signal reconstruit sont clairement visibles.

La construction de la base est une étape importante. Arikan [Ari06] montre que l'utilisation d'ondelettes de Haar sur des données acquises par capture de mouvements donne de piètres résultats. Ceci s'explique d'une part par le fait que les ondelettes de Haar concentrent moins bien l'énergie en présence de signaux continus similaires à ceux rencontrés en capture de mouvements. D'autre part, avec les ondelettes de Haar, la troncation et la quantification introduisent des discontinuités dans le signal reconstruit. De telles discontinuités se traduisent par des coupures dans l'animation, et donc par une dégradation rapide de la qualité lorsque le taux de compression augmente.

Pour solutionner ce dernier problème, illustré à la figure 4.4, nous devons nous assurer que la base d'ondelettes choisie possède de bonnes caractéristiques de continuité. Notre expérience avec les données acquises par capture de mouvements nous porte à croire que, dans certaines situations, un observateur peut être sensible aux discontinuités d'accélération dans une animation. Ceci est confirmé dans une certaine mesure par Reitsma et Pollard [RP03]. Nous devons donc choisir une base d'ordre 4 ou plus (dérivable au moins trois fois).

Nos expérimentations avec différentes bases ont montrées que les ondelettes cubiques interpolantes définies sur un intervalle offrent un excellent compromis entre l'efficacité des algorithmes et la capacité à concentrer l'énergie des fonctions typiquement rencontrées. Pour caractériser entièrement cette base, il suffit de fournir sa transformée en ondelettes directe. Nous procédons à l'aide du schéma de *lifting*.

Le schéma de *lifting* pour la base retenue est illustré à la figure 4.5. Dans cette figure, l'étape split sépare un signal s_j contenant $2^j + 1$ échantillons en $2^{j-1} + 1$ échantillons pairs et 2^{j-1}

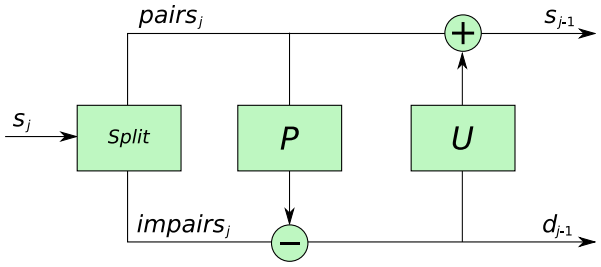


FIG. 4.5 – Les différentes étapes d'une transformée en ondelettes directe réalisée en utilisant le *lifting*. Ce schéma est appliqué récursivement au signal décimé s_{j-1}.

échantillons impairs. Dans l'étape de prédiction $\mathcal{P}$, chaque échantillon impair est prédit par une combinaison linéaire d'échantillons pairs. Dans l'étape de mise à jour $\mathcal{U}$, une combinaison linéaire d'échantillons impairs vient perturber les échantillons pairs. Le résultat est un signal décimé s_{j-1} approximant s_j et un ensemble de coefficients d_{j-1} permettant de reconstruire les détails du signal original. La technique est appliquée récursivement à s_{j-1} jusqu'à $j = 0$. Les détails théoriques reliant cette approche à la théorie des ondelettes sont donnés par Sweldens [Swe98].

Pour terminer la caractérisation de la base, il suffit de spécifier les coefficients utilisés dans les étapes $\mathcal{P}$ et $\mathcal{U}$. Pour ce faire, nous distinguons un certain nombre de cas. Tout d'abord, dans les cas où $j < 3$ nous ne possédons pas assez d'échantillons pairs et impairs pour appliquer un prédicteur cubique. Nous appliquons alors un prédicteur quadratique, linéaire ou constant selon le cas. Ensuite, comme le signal n'est pas défini hors de l'intervalle, nous utilisons des prédicteurs différents pour les premiers et les derniers échantillons. Pour chacun de ces cas, les différentes formes de $\mathcal{P}$ et de $\mathcal{U}$ sont données dans le tableau 4.1. Dans le cas $j \geq 3$, le tableau indique uniquement les ajustements à apporter au début de l'intervalle, des ajustements symétriques doivent être apportés à la fin de l'intervalle. Sweldens [Swe98] montre qu'une construction du genre donne lieu à des ondelettes biorthogonales. Il est à noter que la base résultante n'est pas normalisée, celle-ci peut être normalisée en divisant chaque coefficient de détail d_j par $2^{j/2}$.

La transformation en ondelettes directe donne donc lieu à un algorithme très simple qui s'exécute en temps $\mathcal{O}(n)$, où n est le nombre initial d'échantillons dans le signal. Pour implanter la transformation inverse, il suffit d'observer la réversibilité directe du *lifting scheme*, illustré à la figure 4.6 pour la base d'ondelettes décrite précédemment. Nous avons implanté ces algorithmes en *C++* ainsi qu'à l'aide de *Matlab*.

$$
\begin{array}{l|ll}
j=1 & d_{j-1,0} & = s_{j,1} - \mathcal{P}(s_{j,0}, s_{j,2}) \\
 & & = s_{j,1} - \frac{1}{2}(s_{j,0} + s_{j,2}) \\
 & s_{j-1,0} & = s_{j,0} + \mathcal{U}(d_{j-1,0}) \\
 & & = s_{j,0} + \frac{1}{2} d_{j-1,0} \\
 & s_{j-1,1} & = s_{j,2} + \mathcal{U}(d_{j-1,0}) \\
 & & = s_{j,2} + \frac{1}{2} d_{j-1,0} \\
\hline
j=2 & d_{j-1,0} & = s_{j,1} - \mathcal{P}(s_{j,0}, s_{j,2}, s_{j,4}) \\
 & & = s_{j,1} - \frac{1}{8}(3s_{j,0} + 6s_{j,2} - s_{j,4}) \\
 & d_{j-1,1} & = s_{j,3} - \mathcal{P}(s_{j,0}, s_{j,2}, s_{j,4}) \\
 & & = s_{j,3} - \frac{1}{8}(-s_{j,0} + 6s_{j,2} + 3s_{j,4}) \\
 & s_{j-1,0} & = s_{j,0} + \mathcal{U}(d_{j-1,0}, d_{j-1,1}) \\
 & & = s_{j,0} - \frac{1}{4}(3d_{j-1,0} - d_{j-1,1}) \\
 & s_{j-1,1} & = s_{j,2} + \mathcal{U}(d_{j-1,0}, d_{j-1,1}) \\
 & & = s_{j,2} - \frac{1}{4}(d_{j-1,0} + d_{j-1,1}) \\
 & s_{j-1,2} & = s_{j,4} + \mathcal{U}(d_{j-1,0}, d_{j-1,1}) \\
 & & = s_{j,4} - \frac{1}{4}(-d_{j-1,0} + 3d_{j-1,1}) \\
\hline
j \geq 3 & d_{j-1,0} & = s_{j,1} - \mathcal{P}(s_{j,0}, s_{j,2}, s_{j,4}, s_{j,6}) \\
 & & = s_{j,1} - \frac{1}{16}(5s_{j,0} + 15s_{j,2} - 5s_{j,4} + s_{j,6}) \\
 & d_{j-1,k} & = s_{j,2k+1} - \mathcal{P}(s_{j,2k-2}, s_{j,2k}, s_{j,2k+2}, s_{j,2k+4}) \\
 & & = s_{j,2k+1} - \frac{1}{16}(-s_{j,2k-2} + 9s_{j,2k} + 9s_{j,2k+2} - s_{j,2k+4}) \\
 & s_{j-1,0} & = s_{j,0} + \mathcal{U}(d_{j-1,0}, d_{j-1,1}, d_{j-1,2}, d_{j-1,3}) \\
 & & = s_{j,0} + \frac{1}{32}(35d_{j-1,0} - 35d_{j-1,1} + 21d_{j-1,2} - 5d_{j-1,3}) \\
 & s_{j-1,1} & = s_{j,2} + \mathcal{U}(d_{j-1,0}, d_{j-1,1}, d_{j-1,2}, d_{j-1,3}) \\
 & & = s_{j,2} + \frac{1}{32}(5d_{j-1,0} + 15d_{j-1,1} - 5d_{j-1,2} + d_{j-1,3}) \\
 & s_{j-1,k} & = s_{j,2k} + \mathcal{U}(d_{j-1,k-2}, d_{j-1,k-1}, d_{j-1,k}, d_{j-1,k+1}) \\
 & & = s_{j,2} + \frac{1}{32}(-d_{j-1,0} + 9d_{j-1,1} + 9d_{j-1,2} - d_{j-1,3})
\end{array}
$$

TAB. 4.1 – Valeurs des différents prédicteurs $\mathcal{P}$ et mises à jour $\mathcal{U}$ pour la transformation en ondelettes cubiques interpolantes définies sur un intervalle.

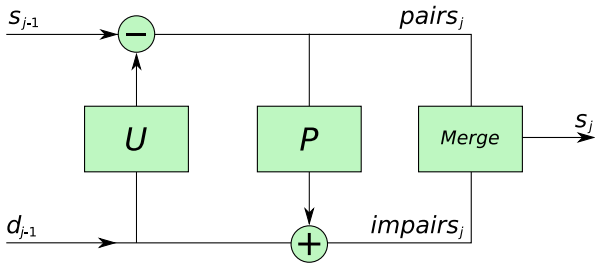


FIG. 4.6 – Les différentes étapes d'une transformée en ondelettes inverse réalisée en utilisant le *lifting scheme*.

4.4 Compression en ondelettes directe

Les notions introduites précédemment nous donnent les outils nécessaires pour développer une compression en ondelettes adaptée aux données acquises par capture de mouvements. La présente section introduit une compression qui s'attarde à réduire les erreurs angulaires. Cette compression, que nous nommons *compression directe*, sera utilisée comme base de comparaison pour les techniques plus évoluées introduites par la suite.

4.4.1 Compression d'un signal unique

Lorsqu'on utilise le schéma de compression simplifié de la figure 4.2(b), le problème de la compression d'un signal peut se poser comme suit : Étant donné r le taux de compression visé ainsi que s un signal contenant n échantillons (parfaitement représentés par n coefficients d'ondelettes), conserver exactement n/r coefficients d'ondelettes de façon à obtenir le signal reconstruit $\hat{s}$ minimisant la distorsion.

Si la distorsion est définie par

$$\sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(s(t)-\hat{s}(t)\right)^2}$$

et que la base d'ondelettes est orthonormale alors on peut facilement montrer que $\hat{s} = \mathrm{trunc}(s, n/r)$. Malheureusement, il n'existe aucune base orthonormale respectant les propriétés désirées de continuité, de symétrie et de support fini. Cependant, on peut vérifier que la base cubique interpolante proposée se rapproche d'une base orthogonale [SS95]. C'est-à-dire que, si ψ_i et ψ_j sont des fonctions de la base, $\langle \psi_i, \psi_j \rangle \approx 1$ quand $i = j$ et $\langle \psi_i, \psi_j \rangle \approx 0$ sinon. À l'instar de beaucoup de techniques de compression basées sur les ondelettes biorthogonales, nous utilisons donc la simplification $\hat{s} \approx \mathrm{trunc}(s, n/r)$.

La constatation précédente nous indique que la meilleure compression (nous omettons l'approximation pour simplifier le texte) sera obtenue en conservant les n/r plus grands coefficients d'ondelettes et en annulant les autres. Cette approche est très générale et peut s'adapter à n'importe quel signal temporel. Par exemple, si on dispose d'un signal vectoriel $\mathbf{d}(t)$, on pourrait appliquer l'approche précédente à chacune des dimensions pour obtenir $\hat{\mathbf{d}}_i(t) = \mathrm{trunc}(\mathbf{d}_i, n/r)$. En général, ce traitement indépendant des signaux, similaire à celui proposé par Arikan [Ari06], s'avère sous-optimal. Par exemple, dans le cas de données acquises par capture de mouvements, la qualité de l'animation reconstruite dépend simultanément de tous les signaux.

4.4.2 Compression de signaux d'animation

Les signaux composant une animation acquise par capture de mouvements donnent donc lieu à une définition légèrement différente du problème de compression. Si l'animation compte k signaux contenant chacun n échantillons, le problème devient : Étant donné r le taux de compression visé et les signaux $\mathbf{d}_i$, conserver kn/r coefficients parmi les représentations en ondelettes de tous les signaux de façon à obtenir les signaux reconstruits $\hat{\mathbf{d}}_i$ minimisant la distorsion.

Pour le moment, attardons-nous à réduire la distorsion angulaire définie par

$$\varepsilon^{\mathtt{A}} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} \left|\mathbf{d}(t) - \hat{\mathbf{d}}(t)\right|^2} \,.$$

On trouve facilement qu'en présence d'une base orthonormale, $\varepsilon^{\mathtt{A}}$ est minimisée en conservant les kn/r plus grands coefficients d'ondelettes parmi tous les kn coefficients. L'approximation introduite précédemment est encore valide et elle nous permet d'appliquer cette conclusion à la base en ondelettes cubique utilisée.

Une fois ces coefficients identifiés, on peut construire un vecteur d'entiers $\mathbf{m}^{\mathtt{D}} \in [0, n]^k$ indiquant le nombre de coefficients à conserver pour chacun des k signaux. Ainsi on peut écrire $\hat{\mathbf{d}}_i = \mathrm{trunc}(\mathbf{d}_i, \mathbf{m}_i^{\mathtt{D}})$.

Cette approche fonctionne bien lorsque tous les signaux utilisent les mêmes unités et ont une importance comparable. Cependant, on sait que les $\mathbf{d}_i$ n'ont pas une importance uniforme. De plus, ces $\mathbf{d}_i$ représentent tous des angles sauf $(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3)$ qui représentent la position de la racine du squelette (voir section 2.2.1). Pour pallier à ce problème, nous avons introduit la mesure de distorsion suivante

$$\varepsilon^{\mathtt{D}} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\sum_{i=1}^{k} \mathbf{w}_i\big(\mathbf{d}_i(t) - \hat{\mathbf{d}}_i(t)\big)^2}$$

qui contient une pondération $\mathbf{w}_i$ indépendante pour chaque degré de liberté. On peut montrer que $\varepsilon^{\mathtt{D}}$ sera minimisée en multipliant tout d'abord chaque coefficient d'ondelettes par le $\mathbf{w}_i$ associé et en choisissant ensuite les kn/r plus grands coefficients pondérés. Les pondérations ne servent qu'à sélectionner les $\mathbf{m}_i^{\mathtt{D}}$, et donc les $\mathbf{w}_i$ n'affectent pas la valeur des coefficients utilisés lors de la compression.

4.5 Sélection optimisée des coefficients

La section précédente a montré comment développer une compression en ondelettes adaptée aux animations acquises par capture de mouvements lorsque le but est de minimiser la distorsion angulaire. Nous montrons maintenant comment cette compression peut être améliorée en présence de mesures de distorsion plus complexes.

4.5.1 Définition d'un espace de recherche

On sait que la distorsion angulaire n'est en général pas une bonne mesure de la fidélité perceptuelle d'une animation compressée. À la section 2.3.3, nous avons argumenté que la distorsion positionnelle constituait une meilleure approximation de la qualité perçue d'une animation. Cette distorsion est définie par

$$\varepsilon^{\mathsf{P}} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\sum_{i=1}^{k^{\mathsf{P}}}\frac{\mathbf{l}_i^{\mathsf{P}}}{\sum_{j=1}^{k^{\mathsf{P}}}\mathbf{l}_j^{\mathsf{P}}}\left(\mathbf{d}_i^{\mathsf{P}}(t)-\hat{\mathbf{d}}_i^{\mathsf{P}}(t)\right)^2}.$$

Les relations non-linéaires entre les valeurs des degrés de liberté $\mathbf{d}_i$ et les positions des joints $\mathbf{d}_i^{\mathsf{P}}$ complexifient grandement le problème de la sélection des coefficients à conserver pour minimiser ε^{P}.

Une première approche consiste à résoudre le problème dans sa forme originale à l'aide d'une recherche exhaustive. Autrement dit, il suffit d'énumérer tous les sous-ensembles de kn/r coefficients parmi les kn coefficients d'ondelettes et de conserver l'ensemble qui minimise ε^{P}. Pour ce problème, la taille de l'espace de recherche est $\binom{kn}{kn/r}$. En général, le nombre d'échantillons n est très grand. Ainsi, la complexité de la recherche croît très rapidement lorsque le taux de compression désiré augmente. En pratique cette approche par force brute s'avère inapplicable pour toute valeur non triviale de r.

Cependant, on peut constater que malgré la non-linéarité de la relation entre les angles et les positions des joints, ε^{P} reste très dépendant de la nature hiérarchique des données. C'est-à-dire que les joints près de la racine ont une influence beaucoup plus grande sur la distorsion positionnelle que les joints près des feuilles de la hiérarchie.

Les facteurs de pondération $\mathbf{w}_i$ que nous avons introduits dans la définition de la distorsion ε^{D} nous permettaient justement de favoriser les degrés de liberté importants. Malheureusement, la technique proposée à la section 2.3.2 pour définir les $\mathbf{w}_i$ est plutôt *ad hoc*, et ceux-ci restent une approximation de l'influence réelle des différents degrés de liberté.

En fait, n'importe quelle définition des $\mathbf{w}_i$ qui ne dépendrait que du squelette est vouée à mal s'adapter à certaines animations. En effet, l'importance d'un degré de liberté ne dépend pas seulement de la nature statique du squelette, mais aussi de la pose dans laquelle il se trouve. La prise en considération des poses présentes dans une animation est donc un élément important dans le choix des $\mathbf{w}_i$.

Pour trouver un bon ensemble de valeurs de $\mathbf{w}_i$, il est donc nécessaire d'évaluer à quel point ils s'adaptent à l'animation à compresser. Or la distorsion positionnelle peut être vue comme une façon simple et efficace d'évaluer la qualité d'une pondération donnée. On peut ainsi poser le problème suivant : Trouver un ensemble de $\mathbf{w}_i$ donnant lieu à la compression minimisant ε^{P}.

La relation entre les $\mathbf{w}_i$ et ε^{P} reste cependant complexe et une solution numérique au problème précédent implique une optimisation coûteuse dans $\mathbb{R}^k$. On peut néanmoins remarquer que le vecteur de réels $\mathbf{w}$ n'influence que le vecteur d'entiers $\mathbf{m}^{\mathsf{D}}$. Ainsi, plutôt que d'optimiser le vecteur $\mathbf{w}$, nous proposons plutôt de chercher à optimiser le nombre de coefficients à conserver lors de la troncation de chaque signal, ceci tout en s'assurant de ne pas modifier le nombre total de coefficients conservés.

En d'autres termes, le problème consiste à trouver k variables discrètes $\mathbf{m}_i^{\mathsf{P}} \in [0, n]$ de façon à ce que les $\hat{\mathbf{d}}_i = \mathrm{trunc}(\mathbf{d}_i, \mathbf{m}_i^{\mathsf{P}})$ minimisent ε^{P}, et ce sous contrainte que $\sum \mathbf{m}_i^{\mathsf{P}} = nk/r$. La taille de l'espace de solutions pour $\mathbf{m}^{\mathsf{P}}$ est donc d'environ $\binom{kn/r}{k}$, ce qui constitue une réduction très importante par rapport au problème original.

Il est important de noter que la solution de ce problème ne permet pas d'identifier les coefficients optimaux à conserver pour obtenir la représentation en ondelettes générale minimisant ε^{P}. Ceci vient du fait que nous ne sommes pas garantis que l'ensemble optimal de coefficients pour chaque signal peut s'exprimer à l'aide de l'opération trunc. Il est en effet possible que, pour certains signaux, de petits coefficients aient une influence importante sur ε^{P}. En pratique, cependant, nous avons constaté que cette technique simplifiée permettait déjà une amélioration importante de la distorsion positionelle.

4.5.2 Algorithme d'optimisation

Il existe plusieurs approches permettant de réaliser l'optimisation du vecteur $\mathbf{m}^{\mathsf{P}}$. Nous proposons ici un algorithme inspiré du recuit simulé et basé sur certaines heuristiques. Cet algorithme, donné à la figure 4.7, s'est avéré efficace en pratique et les vecteurs $\mathbf{m}^{\mathsf{P}}$ sont généralement optimaux, ou très près de l'optimum.

$\mathbf{m}^{\mathsf{P}} \leftarrow \mathbf{m}^{\mathsf{D}}$
$a \leftarrow \lfloor \max_i \{\mathbf{m}_i^{\mathsf{P}}/2\} \rfloor$
while $s \neq 0$ **do**
  $i \leftarrow \text{random}(1, k)$
  $a' \leftarrow \min(a, \mathbf{m}_i^{\mathsf{P}})$
  $\mathbf{m} \leftarrow \mathbf{m}^{\mathsf{P}}$
  $\mathbf{m}_i \leftarrow \mathbf{m}_i - a'$
  Trouver $j \in \{1, ..., k\}$ tel que $\mathbf{m}_j + a'$ minimise ε^{P}
  if $j \neq i$ **then** $\mathbf{m}_j \leftarrow \mathbf{m}_j + a'$, $\mathbf{m}^{\mathsf{P}} \leftarrow \mathbf{m}$
  else if *trop d'erreurs successives* **then** $a \leftarrow \lfloor a/2 \rfloor$
end

FIG. 4.7 : Algorithme d'optimisation de la sélection des coefficients.

L'algorithme débute tout d'abord en estimant le vecteur $\mathbf{m}^{\mathsf{P}}$ comme étant égal au vecteur $\mathbf{m}^{\mathsf{D}}$ obtenu à l'aide de la compression par ondelettes directe présentée à la section 4.4.2. Ensuite, le pas de l'algorithme, a, est initialisé à la moitié du plus grand $\mathbf{m}_i^{\mathsf{P}}$. Ce pas sera graduellement diminué au fur et à mesure de la convergence de l'algorithme.

La boucle principale consiste à transférer l'importance d'un signal vers un autre. Pour ce faire, un signal d'origine i est choisi aléatoirement et son importance $\mathbf{m}_i^{\mathsf{P}}$ est réduite de a. Tous les signaux sont ensuite parcourus de façon à trouver le signal j qui causera la plus importante réduction de ε^{P}. Si ce signal est différent de i, alors on ajoute a à $\mathbf{m}_j^{\mathsf{P}}$ et l'algorithme se poursuit normalement. Si $j = i$, alors l'algorithme enregistre un échec et une nouvelle itération est réalisée en s'assurant de ne pas choisir le même i. Après un certain nombre d'échecs consécutifs, la valeur de a est divisée par 2.

Le nombre d'échecs successifs causant une réduction de a peut être fixé à k de façon à s'assurer que tous les choix possibles de i ont été explorés. En pratique, cependant, nous avons remarqué qu'un nombre d'échecs de l'ordre de $k/5$ donnait des résultats similaires tout en réduisant de façon importante les temps de calcul.

Rien n'empêche d'utiliser cet algorithme avec une mesure de distorsion différente. Cependant, le principe de redistribution des pondérations n'a de sens que si la distorsion est grandement dépendante de la hiérarchie.

4.5.3 Optimisations

Pour accélerer le processus d'optimisation nous avons fait appel à deux optimisations visant à réduire le temps d'évaluation de ε^{P}. Ces deux optimisations nécessitent de conserver une copie du signal $\hat{\mathbf{d}}^{\mathsf{P}}$, indiquant la trajectoire de tous les joints du squelette après compression. Chaque fois que des coefficients d'ondelettes doivent être modifiés dans la représentation de $\hat{\mathbf{d}}$, on met à jour $\hat{\mathbf{d}}^{\mathsf{P}}$.

La première optimisation vient du fait que les fonctions de la base d'ondelettes ont un support fini. Ainsi, un changement à un coefficient d'ondelettes donné a une influence sur un sous-ensemble des échantillons de $\hat{\mathbf{d}}$. On sait donc que seuls les échantillons correspondants de $\hat{\mathbf{d}}^{\mathsf{P}}$ seront modifiés.

La seconde optimisation vient du fait qu'un degré de liberté donné a une influence seulement sur un sous-ensemble des joints du squelette. Ainsi, lorsque $\hat{\mathbf{d}}_i$ est modifié, on peut trouver l'ensemble $\{j_1, j_2, ...\}$ contenant les indices des joints affectés et mettre à jour uniquement les composantes $\hat{\mathbf{d}}^{\mathsf{P}}_{j_1}, \hat{\mathbf{d}}^{\mathsf{P}}_{j_2}, ...$ du signal vectoriel des positions.

La combinaison de ces deux optimisations fait en sorte qu'on limite la mise à jour de $\hat{\mathbf{d}}^{\mathsf{P}}$ à un petit nombre d'échantillons et de dimensions. Ceci accélère de façon importante l'évaluation de ε^{P}.

4.6 Cinématique inverse pour la correction des pieds

La technique précédente, basée sur la distorsion positionnelle, laisse présager qu'une augmentation graduelle du taux de compression est susceptible de produire des problèmes sur n'importe quel joint du squelette. En pratique, cependant, on constate que ces problèmes apparaissent tout d'abord sous la forme d'un glissement des pieds posés au sol. Notre sensibilité particulière aux erreurs se produisant sur les joints en contact avec l'environnement est d'ailleurs bien connue des animateurs et a été relevée par différents chercheurs [KSG02, LM06, IAF06].

Pour pallier à ce problème, une première approche consiste à utiliser une mesure de distorsion positionnelle prenant en considération ces erreurs. Nous avons donc modifié la technique de sélection optimisée des coefficients en la combinant à une métrique de distorsion mettant un poids très important sur les erreurs subies par un pied posé au sol. Malheureusement, la minimisation de telles distorsions implique des ajustements locaux à l'intérieur des signaux de façon à réinsérer les hautes fréquences qui se produisent lors d'un contact. De tels ajustements

font en sorte que certains petits coefficients d'ondelettes ont une influence importante sur la distorsion. Conséquemment, il apparaît que l'opération `trunc` n'est pas assez expressive pour identifier le bon ensemble de coefficients à conserver. De telles mesures de distorsion dépendent trop fortement des signaux eux-mêmes et trop peu de la hiérarchie, et donc ne se prêtent pas à l'optimisation introduite précédemment.

Ikemoto *et al.* [IAF06] ont introduit une technique permettant de détecter et corriger automatiquement les glissements de pieds. Cette méthode, basée sur l'apprentissage d'un classificateur, est particulièrement utile lorsqu'aucune animation de référence n'est disponible. On peut l'utiliser, par exemple, pour corriger des données légèrement bruitées. Dans le cas qui nous occupe, cependant, cette approche risque d'introduire des mouvements qui ne se trouvaient pas dans l'animation originale.

Nous proposons plutôt une méthode qui construit sur les résultats précédents en ajoutant une étape de correction basée sur la cinématique inverse après la reconstruction de l'animation. Le but de cette correction est de replacer les joints erronnés à leur position de contact tirée de l'animation originale. Tout comme Arikan [Ari06], nous ne nous sommes pas attardés au problème de l'identification automatique des joints en contact avec l'environnement et avons préféré corriger la position des pieds, où la plupart des contacts se produisent, pour toute la durée de l'animation.

Nous devons donc stocker trois signaux supplémentaires pour chacun des pieds. Ceci cause une augmentation non négligeable de la taille des données considérant que l'animation complète compte 62 signaux. Il apparaît donc nécessaire de compresser aussi ces nouveaux signaux.

La compression en ondelettes de ces signaux cause cependant un problème. En effet, la position des pieds peut couvrir une très grande plage de valeurs qui risque de nécessiter un grand nombre de coefficients d'ondelettes pour être compressée précisément. Cette précision est néanmoins requise à défaut de quoi la qualité de l'animation reconstruite risque d'être réduite de façon dramatique. Ceci est dû à la cinématique inverse qui pourrait tenter d'ajuster un grand nombre de degrés de liberté de façon à réaliser la contrainte imprécise.

Nous contournons ce problème en considérant plutôt des signaux contenant la différence entre la position des pieds issue de la compression en ondelettes et leur position originale. Ces différences sont généralement très près de 0 et couvrent un domaine beaucoup plus petit que les positions elles-mêmes. Elles sont ainsi beaucoup plus faciles à encoder en utilisant peu de coefficients d'ondelettes, et elles éliminent le risque de dégrader la qualité de l'animation.

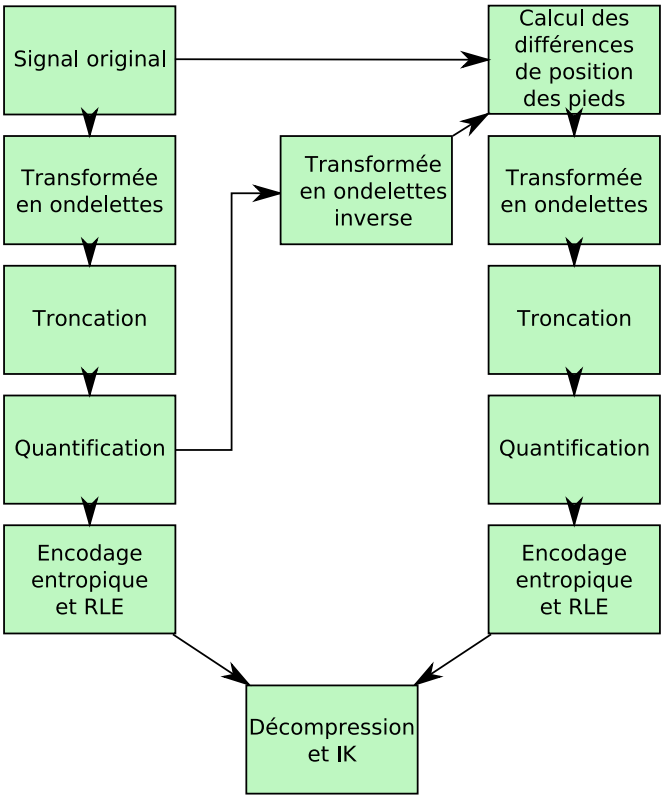


FIG. 4.8 – Les différentes étapes de la compression suivies d'une étape de correction par *IK*. Le taux de compression pour la première troncation (à gauche) est r et celui de la seconde troncation (à droite) est $r^{\mathtt{IK}}$.

De plus, ces signaux contiennent essentiellement les hautes fréquences se produisant lors des contacts et qui avaient été éliminées lors de la compression des degrés de liberté. Nous pouvons donc espérer que ces détails pourront seront réintroduits dans l'animation reconstruite.

Le fait d'encoder les différences nous oblige cependant à réaliser la compression par étapes, illustrées à la figure 4.8. On doit d'abord produire l'animation reconstruite $\hat{\mathbf{d}}$, puis calculer les signaux encodant la différence de position des pieds entre $\mathbf{d}$ et $\hat{\mathbf{d}}$, et finalement utiliser la compression en ondelettes directe (section 4.4.2) sur ces 6 nouveaux signaux. Il est donc nécessaire de préciser un nouveau taux $r^{\mathtt{IK}}$ régissant la compression des signaux des pieds. Ainsi, le taux final de compression dépend simultanément de r et de $r^{\mathtt{IK}}$.

La dernière étape imposée par cette technique consiste en l'application de la cinématique inverse (*IK*) sur l'animation reconstruite. Il existe plusieurs solutionneurs de *IK* différents, offrant chacun leurs avantages et leurs inconvénients [Wel93, Fěd03, MM05]. Dans le cas qui nous occupe, il est important de choisir une technique retournant une solution rapprochée de la

pose de départ. Ceci dans le but de s'assurer d'apporter le moins de changements possible au squelette lorsque l'erreur est petite.

Dans notre implantation, nous avons opté pour une technique de *cyclic coordinate descent* (*CCD*) [Wel93] où le pas de recherche a été réduit de façon à favoriser les solutions se rapprochant de la pose initiale. La *CCD* est reconnue pour ses piètres performances en terme de temps de calcul, mais nous l'avons retenue pour sa simplicité et sa robustesse. Une application où l'efficacité de la décompression est cruciale aurait avantage à utiliser une technique alternative possédant des propriétés similaires, comme des méthodes basées sur le jacobien [Fěd03, Wel93].

Bien que nous nous soyons limités à l'encodage des pieds, rien ne nous empêche d'utiliser cette même technique pour corriger tous les joints entrant en contact avec l'environnement. Naturellement, la qualité finale de la compression risque de diminuer lorsque plusieurs joints sont encodés de la sorte. En général, cependant, les contacts sont localisés dans le temps et il est possible d'encoder les signaux de correction uniquement au moment où ils se produisent. Les instants où se produisent les contacts peuvent être manuellement annotés dans l'animation ou ils peuvent être détectés automatiquement à l'aide d'une technique similaire à celle proposée par Ikemoto *et al.* [IAF06].

4.7 Quantification et encodage

Jusqu'à maintenant, nous avons analysé les différentes compressions suivant le schéma simplifié de la figure 4.2(b). Cette simplification est utile pour l'évaluation du comportement de la compression, mais l'ajout des étapes de quantification et d'encodage, illustrées à la figure 4.2(a), permet d'augmenter substantiellement le taux de compression en ne diminuant que peu la qualité de l'animation reconstruite. Nos principales contributions se trouvant au niveau des techniques de troncation, nous avons opté pour une quantification et un encodage simples. La même quantification et le même encodage sont appliqués à toutes les compressions discutées précédemment.

Chaque coefficient d'ondelettes est tout d'abord quantifié uniformément sur 16 bits. Un certain nombre de codes de 16 bits sont réservés pour l'encodage RLE décrit plus bas. Pour chaque signal, l'intervalle de quantification est identifié en prenant les valeurs minimales et maximales des coefficients d'ondelettes. Ces valeurs sont stockées en format à virgule flottante 32 bits dans le fichier compressé. Pour minimiser l'impact de ces valeurs, nous utilisons un même intervalle

de quantification pour tous les signaux représentant des angles. Un autre intervalle de quantification unique est utilisé pour les signaux représentant les différences de position des pieds.

Grâce à la troncation, cette quantification donne lieu à de nombreuses occurences de zéros. Dans la grande majorité des cas, les coefficients nuls affectent les fonctions d'ondelettes les plus étroites. Nous exploitons donc ce fait en ordonnant les coefficients de la fonction d'ondelettes de la plus large à la plus étroite. En se référant à la figure 4.5, cette ordonnancement devient : $s_{0,0}, s_{0,1}, d_{0,0}, d_{1,0}, d_{1,1}, d_{2,0}, ...$

La liste ordonnée des coefficients quantifiés contient donc généralement de longues séquences de zéros consécutifs. Pour profiter de ces répétitions, nous réservons 255 codes de 16 bits pour représenter des séquences de 2 à 256 zéros. Un autre code est réservé pour indiquer que tous les coefficients restants sont nuls.

Malgré cet encodage *RLE*, l'entropie de fichier reste assez faible. On peut donc augmenter encore légèrement le taux de compression en utilisant un encodeur entropique. Plusieurs choix sont ici possibles, incluant un encodage de Huffman [Huf51] ou un encodage arithmétique [Ris76, RL79]. Nous ne nous sommes pas penchés sur le choix du meilleur encodeur possible mais avons néanmoins étudié l'effet d'un encodage Ziv-Lempel [ZL77, ZL78] tel qu'implanté dans le logiciel libre *gzip*, ceci de façon à estimer la compression maximale pouvant être atteinte par notre technique.

4.8 Résultats

Les différentes techniques de compression présentées précédemment ont été testées sur un grand nombre de mouvements tirés de la *CMU Motion Capture Database* [CMU], une banque publique d'animations acquises par capture de mouvements accessible sur internet. Tous les squelettes utilisés dans ces animations partagent une même hiérarchie et les mêmes degrés de liberté. De plus, toutes les animations considérées sont échantillonnées à 120 Hz.

Cette section discute des temps d'exécution requis pour la compression et la décompression, des taux de compression obtenus et de la qualité des animations reconstruites. Cette dernière facette est probablement celle qui représente le plus grand défi. Tout d'abord parce que l'évaluation de la qualité d'une animation dépend fortement de l'application qu'on souhaite en faire. Comme nous l'avons mentionné au début de ce chapitre, nous nous intéressons plus particulièrement à la façon dont un observateur humain jugerait la qualité de l'animation reconstruite.

Malgré que ceci circonscrive plus précisément le problème, il n'en reste pas moins que l'évaluation de la qualité perceptuelle d'une animation n'est pas une tâche aisée. En animation vidéo, plusieurs chercheurs se sont intéressés au problème et un certain nombre de résultats sont connus [van04, Lub97, WS98]. Peu de ces résultats sont applicables au monde bien différent qu'est l'animation 3D, où l'angle de vue n'est pas déterminé *a priori*. En effet, il est facile de présumer que la qualité perçue d'une animation dépendra, entre autres, de la proximité de la caméra et une métrique de qualité perceptuelle applicable en animation 3D devrait considérer ce facteur, absent des études en animation traditionnelle. D'autres facteurs peuvent aussi jouer un rôle important, comme les occultations, la présence d'ombres projetées, la qualité du maillage, l'utilisation de vêtements animés, etc.

En ce qui concerne les animations acquises par capture de mouvements, le problème est encore plus complexe. En effet, non seulement le point de vue est inconnu, mais souvent on ne connaît ni l'environnement dans lequel l'animation sera exécutée, ni le maillage 3D qui sera contrôlé par l'animation. Ainsi, en l'absence d'une métrique capable de prendre en considération ces différents paramètres, les recherches en compression d'animations utilisent généralement la combinaison d'une évaluation numérique grossière et d'une évaluation qualitative subjective résultant de l'observation des résultats. Nous adoptons cette même approche et utilisons la métrique de distorsion ε^{P} introduite précédemment.

Pour l'évaluation visuelle, nous avons généré les animations reconstruites sur un maillage simple représentant chacun des os du squelette. Un tel maillage, illustré à la figure 4.9, se rapproche des représentations utilisées par beaucoup de logiciels d'animation commerciaux. Pour permettre au lecteur d'évaluer lui-même la qualité des animations reconstruites, des séquences vidéo sont disponibles sur internet [Bea07a]. Dans ces séquences vidéo, la position de la caméra a été choisie de façon à bien montrer l'ensemble du mouvement.

Par souci de simplicité, nous avons procédé à la collecte et à l'évaluation des résultats sur 15 animations spécifiques. Cependant, nous avons observé des résultats similaires sur les autres animations auxquelles nous avons appliqué les compressions décrites précédemment. Les 15 animations ont été retenues pour leur variété autant en termes des types de mouvements représentés que de la durée des séquences. La plus courte contient 148 échantillons (environ une seconde) et la plus longue en contient 5423 (45 secondes). La liste et les caractéristiques de ces 15 animations sont données au tableau 4.2. Dans la suite de ce chapitre, nous donnons des résultats spécifiques pour une animation courte "09_01.amc" et une animation moyenne

"49_02.amc". Nous donnons aussi les taux de compression et les erreurs moyennes obtenues sur l'ensemble des animations de référence.

4.8.1 Animations de référence

En vue d'évaluer des taux de compression, nous devons tout d'abord décrire précisément le format des fichiers de référence. Notons tout d'abord que les fichiers originaux disponibles sur la *CMU Motion Capture Database* sont stockés dans le format AMC (*Acclaim Motion Capture data*). Ce format stocke la liste ordonnée de tous les échantillons (*frames*). Pour chacun de ces échantillons, le fichier liste le nom de tous les joints et la valeur de tous les degrés de liberté associés à ce joint. Le nom des joints est une simple chaîne de caractères *ASCII* et les degrés de liberté sont des nombres décimaux à virgule flottante exprimés en *ASCII* en utilisant 6 chiffres significatifs.

Il va sans dire que le format AMC a été tout d'abord conçu pour faciliter la lisibilité du fichier en permettant de l'explorer et de le manipuler à l'aide d'un simple éditeur de texte. À cet effet, le format contient de nombreuses informations redondantes comme le nom des joints répété à chaque échantillon. De plus, les degrés de liberté sont stockés en *ASCII* alors qu'ils peuvent être exprimés dans un format binaire de façon beaucoup plus compacte sans engendrer de pertes de précision.

Le format AMC génère donc des fichiers de très grande taille qui ne constituent pas, à notre avis, une référence adéquate pour le calcul des taux de compression. Nous décrivons donc maintenant un nouveau format permettant d'exprimer ces données de façon beaucoup plus compacte tout en n'engendrant aucune perte de précision. Ce nouveau format constituera la référence utilisée par la suite pour toutes les évaluations des taux de compression.

Dans un premier temps, nous proposons d'éliminer l'information redondante en omettant tout simplement de stocker le nom des joints. Cette simplification est adéquate étant donné que ces joints ainsi que leurs degrés de liberté sont contenus dans un fichier de description de squelette, aussi disponible sur la *CMU Motion Capture Database* en format ASF (*Acclaim Skeleton Format*). Ces fichiers de description de squelette sont compacts, indépendants des animations, et sont utilisés avec de nombreuses animations différentes. Ainsi, en pratique, ils ont un impact négligeable sur la taille d'une animation. Nous les ignorons donc pour le calcul du taux final de compression.

La seconde optimisation que nous proposons est le stockage des valeurs des degrés de li-

Nom de fichier	Description	Nombre d'échantillons	Taille (ko) AMC	Binaire	*gzip*
01_02.amc	Grimpe	4346	3386	1052.6	949.7
05_13.amc	Danse	1095	852	265.2	239.2
06_13.amc	Basketball	4905	3825	1187.9	1072.2
08_01.amc	Marche	277	217	67.1	60.4
08_10.amc	Marche	275	215	66.6	60.1
09_01.amc	Course	148	116	35.8	32.4
09_09.amc	Course	152	119	36.8	33.1
09_12.amc	Marche complexe	1918	1497	464.5	420.0
13_17.amc	Boxe	4840	3759	1172.2	1060.7
13_29.amc	Exercices	4592	3592	1112.1	1002.0
14_02.amc	Boxe	5423	4230	1313.4	1178.1
16_35.amc	Jogging	162	126	39.2	35.3
49_02.amc	Saut sur un pied	2085	1640	505.0	453.3
87_01.amc	Saut jambe tendue	616	481	149.2	134.3
88_05.amc	Salto arrière	149	116	36.1	32.5

TAB. 4.2 – Caractéristiques des animations compressées. Les noms de fichier se réfèrent à la *CMU Motion Capture Database* [CMU].

berté directement en binaire. Pour ce faire, nous utilisons une représentation binaire à virgule flottante du type IEEE 754. Le format final contient donc une liste de k nombres de 32 bits répétée pour chaque échantillon. Le tableau 4.2 liste les différents fichiers considérés, le nombre d'échantillons qu'ils contiennent, leur taille en format AMC et leur taille en format de référence. À titre de comparaison, on indique aussi la taille du fichier binaire compressé sans pertes en utilisant l'encodage Ziv-Lempel implantée par le logiciel *gzip*.

4.8.2 Quantification

Avant d'étudier le comportement des compressions introduites plus haut, nous présentons une première compression avec pertes simple agissant comme base comparative. Nous avons en effet remarqué que la grande précision de la représentation à virgule flottante est souvent inutile pour les données qui nous intéressent. Ceci est corroboré par le fait que le format AMC ne conserve que 6 chiffres décimaux significatifs, une représentation moins précise que les 24 bits significatifs du format IEEE 754. Nous proposons donc de quantifier les données de façon à stocker chaque scalaire à l'aide d'un moins grand nombre de bits.

Le paramètre de contrôle d'une compression par quantification est le nombre de casiers (*bins*) utilisés. Étant donné un taux de compression visé $r^{\mathtt{Q}}$, chaque scalaire doit être stocké à

l'aide de $32/r^{\mathtt{Q}}$ bits, donc le nombre de casiers à utiliser est de $2^{32/r^{\mathtt{Q}}}$. De plus, les frontières des domaines de quantification doivent aussi être spécifiées. Pour une animation donnée, nous utilisons quatre domaines de quantification différents : trois pour les degrés de liberté $\mathbf{d}_1, \mathbf{d}_2$ et $\mathbf{d}_3$ correspondant à la position de la racine et un autre pour tous les autres degrés de liberté angulaires. Les frontières de ces domaines sont exprimées à l'aide de huit nombres binaires à virgule flottante IEEE 754 contenus dans l'en-tête du fichier. Ces domaines sont déterminés en identifiant les valeurs minimales et maximales des différents degrés de liberté auxquels ils s'appliquent.

Le format compressé contient ainsi un en-tête de 256 bits (8×32 bits) suivi d'une liste de k nombres de $32/r^{\mathtt{Q}}$ bits répétée pour chaque échantillon. Les taux de compression et les distorsions $\varepsilon^{\mathtt{P}}$ obtenus à l'aide de cette technique sont comparés aux autres formes de compression au tableau 4.3 et à la figure 4.10.

4.8.3 Sous-échantillonnage

Le sous-échantillonnage constitue un second type de compression simple pouvant servir de base comparative. La compression par sous-échantillonnage consiste à réduire le nombre d'échantillons présents dans l'animation de façon uniforme sur toute l'animation.

Le seul paramètre guidant cette compression est le taux de sous-échantillonnage $r^{\mathtt{SE}}$, indiquant le rapport entre le nombre d'échantillons de l'animation originale et le nombre d'échantillons de la version compressée. Par exemple, pour $r^{\mathtt{SE}} = 2$, seuls les échantillons pairs seraient conservés.

En général, pour les valeurs de $r^{\mathtt{SE}}$ non entières, les échantillons de l'animation compressée ne correspondent pas à des échantillons de l'animation originale. Dans ce cas, nous effectuons une interpolation cubique des échantillons originaux.

Le format de stockage est le même que le format de référence, soit 32 bits par scalaire. Les taux de compression et les distorsions $\varepsilon^{\mathtt{P}}$ obtenus à l'aide de cette technique sont comparés aux autres formes de compression au tableau 4.3 ainsi qu'à la figure 4.10. Notons qu'il serait possible de combiner le sous-échantillonnage à la quantification introduite précédemment pour obtenir de meilleurs taux de compression, mais nous n'avons pas spécifiquement étudié ce type de compression mixte.

4.8.4 Analyse des composantes principales

Nous introduisons une troisième compression simple utilisée elle aussi comme base comparative. Celle-ci exploite la corrélation entre les degrés de liberté et ignore la cohérence temporelle. Cette technique consiste à réaliser une analyse des composantes principales (*PCA*) sur l'ensemble des n échantillons. Ceci résulte en l'identification d'une base orthonormale dans laquelle les vecteurs de base sont ordonnés de la dimension la plus importante à la moins importante. L'importance d'un vecteur de base est mesurée par la variance des échantillons projetés sur ce vecteur.

Si la dimension initiale est de k, la compression consiste à projeter chaque échantillon sur les $k^{\texttt{PCA}} \leq k$ vecteurs de base les plus importants tels que déterminés par *PCA*. On détermine généralement $k^{\texttt{PCA}}$ en spécifiant le pourcentage d'énergie[2] à conserver après la projection. Cependant, de façon à employer un paramètre plus direct et comparable avec ceux utilisés par les autres types de compression, nous spécifions plutôt $r^{\texttt{PCA}}$ indiquant le ratio $k/k^{\texttt{PCA}}$ souhaité. Si ce ratio ne peut être atteint exactement avec $k^{\texttt{PCA}}$ entier, alors on choisit le $k^{\texttt{PCA}}$ entier donnant lieu au ratio se rapprochant le plus de $r^{\texttt{PCA}}$. Rappelons que $k = 62$ pour les squelettes que nous utilisons.

Étant donnés n échantillons de départ en k dimensions, le résultat de cette compression est un ensemble de n échantillons en $k^{\texttt{PCA}}$ dimensions ainsi qu'une matrice de taille $k \times k^{\texttt{PCA}}$ permettant de transformer un vecteur projeté vers l'espace original.

Chaque entrée de la matrice $k \times k^{\texttt{PCA}}$ est encodée sur 32 bits en format IEEE 754 en entête du fichier compressé. Le choix d'un format à virgule flottante vient de la sensibilité de cette matrice sur la reconstruction ainsi que de notre difficulté à trouver un bon domaine de quantification. Des analyses plus poussées pourraient probablement permettre de réduire le coût de stockage de cet en-tête, mais nous ne nous sommes pas attardés à ce problème dans le cadre de nos recherches.

Dans le cas de la compression par *PCA*, nous avons évalué qu'une quantification des échantillons sur 16 bits n'introduisait pas d'erreurs significatives. Ainsi, suite à la matrice, l'en-tête contient $2k^{\texttt{PCA}}$ nombres à virgule flottante indiquant les frontières des domaines de quantification des n échantillons projetés. Contrairement au format introduit à la section 4.8.2, nous utilisons ici un domaine de quantification distinct pour chaque dimension. La raison en est

[2] L'énergie correspondant à un vecteur de base est égale à la variance des données projetées sur ce vecteur, ou encore à la valeur propre associée à ce vecteur lors de l'application de la *PCA*.

que ces dimensions sont des agrégats combinant des degrés de liberté positionnels et angulaires et qu'ils ont donc potentiellement des domaines très différents les uns des autres. Il est important de noter que, pour des animations comptant peu d'échantillons, la taille de cet en-tête est non négligeable et peut diminuer sensiblement le taux de compression.

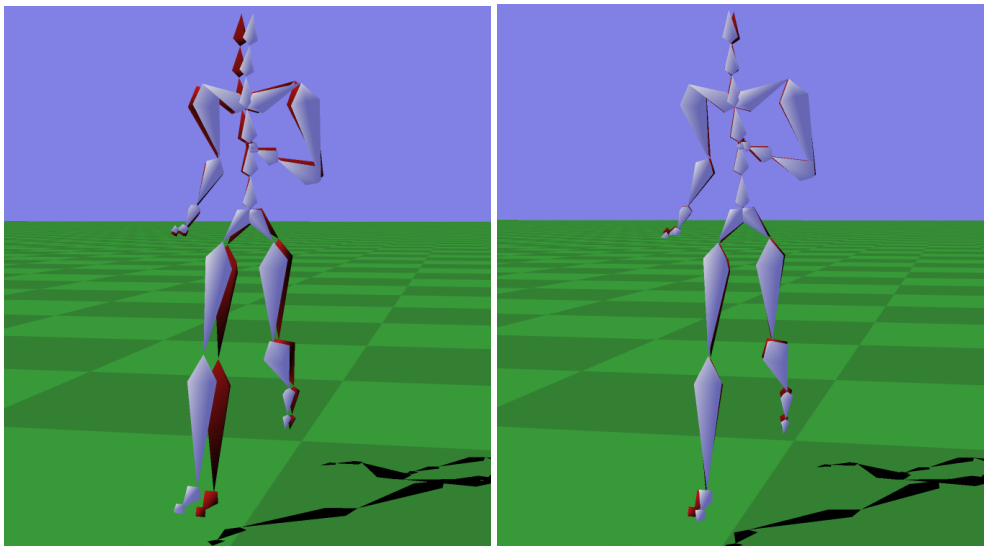
Suite à l'en-tête, chacun des n échantillons projetés est ensuite encodé sous la forme d'une liste de $k^{\mathtt{PCA}}$ nombres quantifiés sur 16 bits. Le tableau 4.3 et la figure 4.10 comparent les résultats obtenus avec la compression par *PCA* aux autres types de compression.

Une analyse visuelle des résultats permet de constater que, pour une même distorsion positionelle ε^{P}, les animations obtenues suite à une compression *PCA* semblent présenter une qualité inférieure à celles obtenues par les autres types de compression. Ceci démontre les limitations de cette mesure de distorsion pour l'évaluation de la qualité perceptuelle. Malgré cela, pour une valeur donnée de ε^{P}, cette technique présente des taux de compression bien inférieurs aux techniques alternatives. Ceci nous conforte dans l'idée que, pour les données qui nous occupent, la cohérence temporelle offre de meilleures opportunités de compression que la corrélation des degrés de liberté.

4.8.5 Compression en ondelettes directe

Nous analysons maintenant les résultats obtenus lors de l'utilisation de la compression en ondelettes directe décrite à la section 4.4. Le format d'un fichier d'animation utilisant la compression en ondelettes directe contient en en-tête huit nombres à virgule flottante indiquant les frontières des domaines de quantification : six pour les domaines des positions 3D de la racine et deux pour les domaines de tous les autres degrés de liberté angulaires. Le reste du fichier contient k signaux, chacun composé de n coefficients d'ondelettes. Ces coefficients sont quantifiés et stockés en utilisant l'encodage *RLE* décrit à la section 4.7.

Les taux de compression et les résultats sont présentés au tableau 4.3 et à la figure 4.10. On remarque que, pour un même taux de compression, la compression en ondelettes directe engendre de plus grandes distorsions positionnelles ε^{P} que la sélection optimisée des coefficients. Lorsqu'on observe les différentes animations, ceci se traduit par des erreurs perceptuellement visibles pour des taux de compression supérieurs à environ 20:1. Pour des taux de compression inférieurs, cette technique produit généralement une animation compressée non distinguable de l'originale. Les temps de compression excluant $gzip$ sont de 50μs par échantillon en moyenne. La décompression, quant à elle, s'exécute en moyenne en 30μs par échantillon, ce qui en théorie



(a) (b)

FIG. 4.9 – Comparaison visuelle entre (a) la compression en ondelettes directe et (b) la sélection optimisée des coefficients pour un taux de compression de 30:1. La pose du mouvement compressé est superposée à celle du mouvement original, visible en rouge. Image tirée de [BPv07b].

pourrait permettre de décompresser plus de 200 animations en temps réel.

4.8.6 Sélection optimisée des coefficients

Nous avons ensuite étudié l'effet de la sélection optimisée de coefficients proposée à la section 4.5. Le format de fichier utilisé ici est identique à celui employé lors de la compression en ondelettes directe. L'unique différence vient du fait que les coefficients d'ondelettes retenus ont été ajustés de façon à minimiser la distorsion positionnelle ε^{P}.

Nous présentons les résutats au tableau 4.3 et à la figure 4.10. On voit facilement que, pour une valeur donnée de ε^{P}, la sélection optimisée des coefficients permet d'obtenir des taux de compression significativement supérieurs à ceux obtenus avec toutes les techniques de compression précédentes. La figure 4.9 compare, pour un même taux de compression, les résultats obtenus avec la compression en ondelettes directe et la sélection optimisée des coefficients. Des comparaisons semblables sont aussi visibles dans la séquence vidéo accompagnant cette thèse [Bea07a].

Suite à une inspection visuelle, nous avons remarqué qu'aucun artéfact n'était visible avec $\varepsilon^{\mathsf{P}} \leq 0.5$. Ainsi, en moyenne, cette technique de compression peut atteindre des taux de compression de 25:1 sans pertes apparentes. Ces taux peuvent grimper jusqu'à 40:1 pour des

ε^{P}	Quantification		Sous-éch.		*PCA*		Ond. directes		Ond. optimisées	
1.4	7.44	(4.8:1)	2.91	(12.3:1)	6.02	(6.0:1)	1.27	(28:1)	0.75	(48:1)
0.96	7.87	(4.6:1)	3.66	(9.8:1)	6.49	(5.5:1)	1.51	(24:1)	0.94	(38:1)
0.58	8.99	(4.0:1)	5.05	(7.1:1)	7.67	(4.7:1)	2.31	(16:1)	1.24	(29:1)
0.26	10.1	(3.5:1)	7.97	(4.5:1)	9.42	(3.8:1)	3.69	(9.7:1)	2.18	(16:1)
0.08	11.8	(3.0:1)	21.1	(1.7:1)	11.5	(3.1:1)	8.74	(4.1:1)	5.08	(7.0:1)

(a) 09_01.amc

ε^{P}	Quantification		Sous-éch.		*PCA*		Ond. directes		Ond. optimisées	
0.67	142	(3.6:1)	52.6	(9.6:1)	153	(3.3:1)	18.6	(27:1)	9.97	(51:1)
0.45	154	(3.3:1)	64.7	(7.8:1)	157	(3.2:1)	24.8	(20:1)	13.5	(37:1)
0.29	167	(3.0:1)	85.6	(5.9:1)	164	(3.1:1)	33.8	(15:1)	18.4	(27:1)
0.14	178	(2.8:1)	136	(3.7:1)	168	(3.0:1)	62.7	(8.1:1)	32.9	(15:1)
0.05	205	(2.5:1)	337	(1.5:1)	177	(2.9:1)	135	(3.7:1)	79.2	(6.4:1)

(b) 49_02.amc

TAB. 4.3 – Taille (en ko) et taux de compression des fichiers compressés à l'aide de différentes techniques de façon à obtenir une distorsion donnée ε^{P} (en cm). Le tableau présente deux animations caractérisant bien les résultats obtenus sur l'ensemble des fichiers de référence.

séquences plus longues. Naturellement, la distorsion ε^{P} visée dépend du contexte. Il peut être souhaitable d'atteindre une valeur plus faible pour un point de vue rapproché de l'animation ou, au contraire, une valeur plus grande pour un point de vue éloigné. En général, lorsqu'on augmente le taux de compression, le premier artéfact visible vient du glissement des pieds.

Le nombre d'itérations et les temps requis pour la compression à l'aide de la sélection optimisée des coefficients sont donnés au tableau 4.4. Le processus de décompression est le même que pour la compression en ondelettes directe et s'effectue donc aussi à $30\mu\mathrm{s}$ par échantillon lorsque *gzip* n'est pas utilisé.

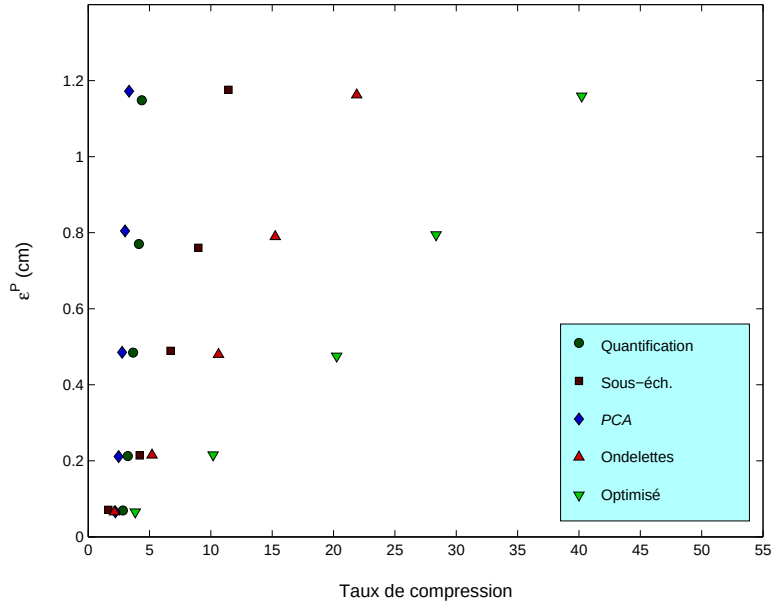
Taux	Temps (ms)	Nb. itér.
50:1	133	264
40:1	214	415
30:1	254	496
15:1	293	542
7:1	327	669

(a) 09_01.amc

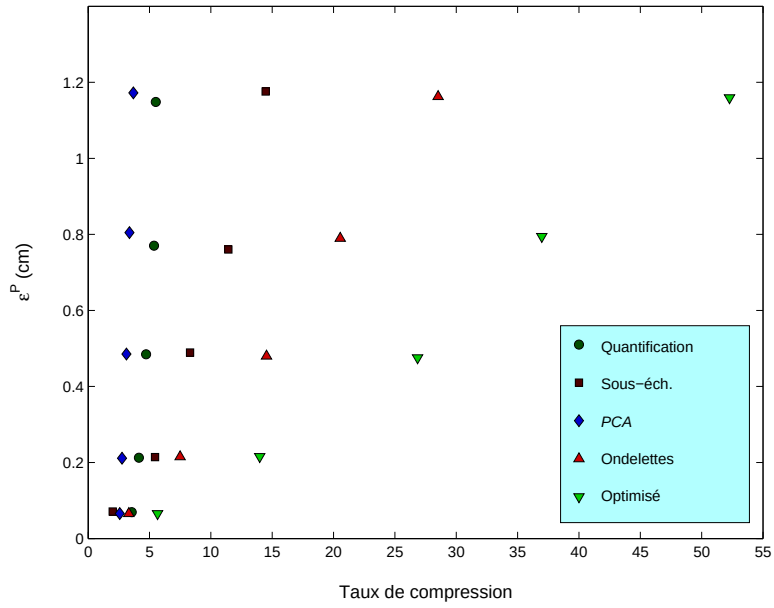
Taux	Temps (ms)	Nb. itér.
50:1	308	247
40:1	312	243
30:1	391	311
15:1	456	369
7:1	483	380

(b) 49_02.amc

TAB. 4.4 – Temps d'exécution par échantillon et nombre d'itérations de l'algorithme de sélection optimisée des coefficients.



(a) Sans *gzip*



(b) Avec *gzip*

FIG. 4.10 – Graphe de taux-distorsion (*rate-distortion*) moyen sur les 15 animations de référence pour les différentes techniques de compression proposées. En (b) un encodage entropique supplémentaire a été appliqué à l'aide de *gzip*.

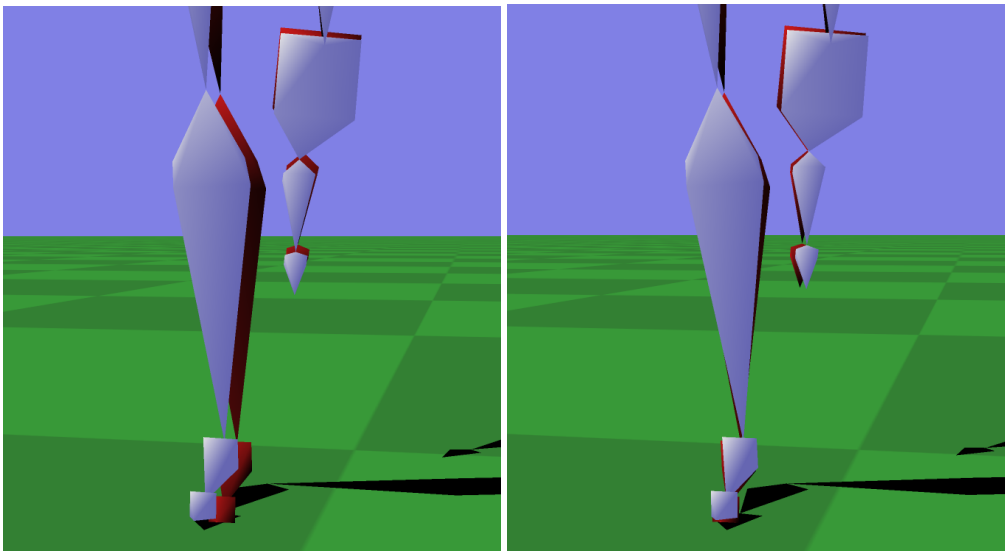
4.8.7 Cinématique inverse pour la correction des pieds

Comme mentionné précédemment, lorsqu'on augmente le taux de compression de la sélection optimisée des coefficients le premier artéfact visible qui apparaît provient du glissement des pieds sur le sol. Pour corriger ce type de problème, nous avons proposé à la section 4.6 une technique de correction par cinématique inverse. Le format d'un fichier faisant appel à ce type de correction est identique à celui décrit précédemment pour la compression en ondelettes, à la différence que six nouveaux signaux doivent être stockés. Le coefficient d'ondelettes de ces signaux sont quantifiés et encodés par *RLE* comme décrit à la section 4.7.

La nécessité de stocker ces signaux supplémentaires fait en sorte que, pour une même taille de fichier, la correction par cinématique inverse doit allouer moins de coefficients d'ondelettes aux degrés de liberté que les techniques précédentes. Ceci se traduit par une légère augmentation de $\varepsilon^{\mathtt{P}}$ lorsque la correction par cinématique inverse est utilisée. Cependant, cette correction résout le principal problème résultant de la sélection optimisée des coefficients. Ainsi, une inspection visuelle nous a permis de constater que la correction par cinématique inverse permettait d'atteindre des taux de compression plus élevés que les compressions précédentes sans introduire d'artéfacts visibles. Ceci est visible, par exemple, à la figure 4.11, qui montre que, pour le même taux de compression, la correction peut réduire considérablement le glissement des pieds en ajoutant seulement des distorsions mineures au reste de l'animation. Ceci peut aussi être observé dans la séquence vidéo accompagnant cette thèse [Bea07a].

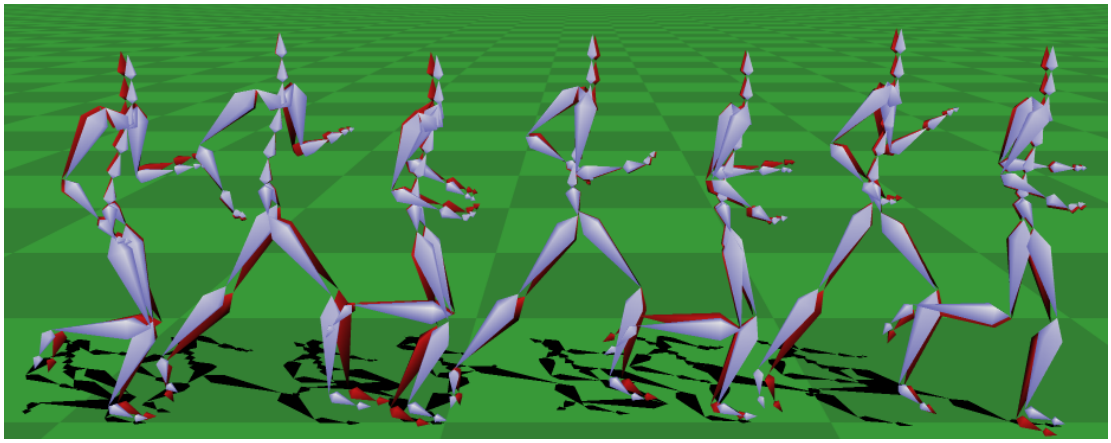
Pour les 15 séquences de test, la correction par cinématique inverse a permis d'atteindre des taux de 35:1 sans artéfacts apparents. Nous avons évalué ceci en comparant visuellement les mouvements résultant de la correction des pieds à ceux obtenus à l'aide des techniques de compression en ondelettes lorsque $\varepsilon^{\mathtt{P}} = 0.5$. Les paramètres utilisés pour obtenir les résultats présentés dans la séquence vidéo sont $r = 30$ et $r^{\mathtt{IK}} = 16$. La figure 4.12 compare les résultats d'une compression en ondelettes standard à une compression utilisant la correction par cinématique inverse lorsqu'un taux de compression agressif est utilisé.

Le temps supplémentaire requis pour extraire les six signaux de correction est de 45μs par échantillon en moyenne. Le temps de décompression moyen est de 300μs par échantillon. Ce nombre important s'explique essentiellement par l'utilisation d'un solveur de cinématique inverse *CCD* (*cyclic coordinate descent*). Ce type de solveur est simple et robuste, mais il est relativement inefficace [Wel93]. Des temps de décompression beaucoup plus courts pourraient être obtenus en utilisant un solveur mieux adapté à ce genre d'application.

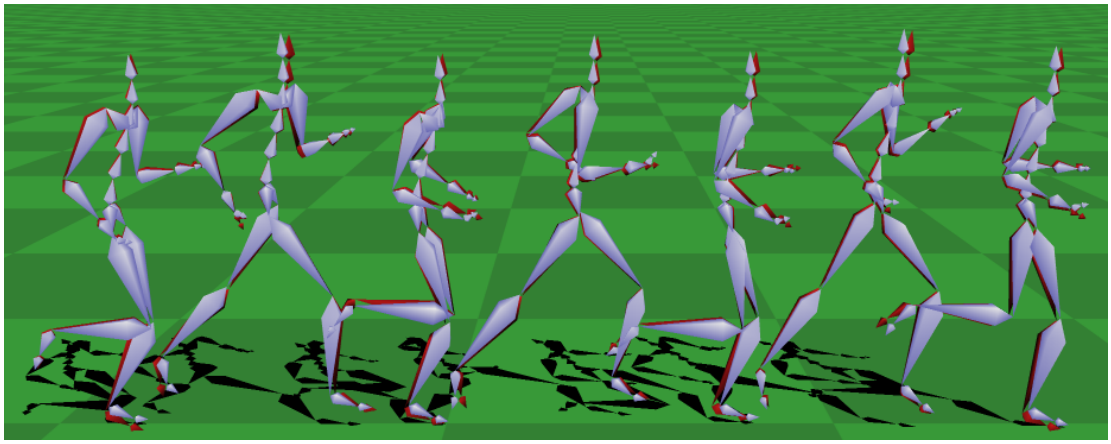


(a) (b)

FIG. 4.11 – Comparaison visuelle entre (a) la compression avec sélection optimisée des coefficients et (b) la compression avec correction des pieds par cinématique inverse pour un taux de compression de 35:1. La pose du mouvement compressé est superposée à celle du mouvement original, visible en rouge. En (a) la différence entre le mouvement original et le mouvement compressé se traduit par une oscillation rapide et un glissement perceptible du pied au cours de l'animation. Image tirée de [BPv07b].



(a)



(b)

FIG. 4.12 – Comparaison visuelle entre (a) la compression en ondelettes directe et (b) la compression avec correction des pieds par cinématique inverse pour un taux de compression agressif de 48:1. Les poses du mouvement compressé sont superposées à celles du mouvement original, visibles en rouge. Image tirée de [BPv07b].

4.9 Conclusion

Dans ce chapitre, nous nous sommes principalement intéressés à exploiter les cohérences temporelles à petite échelle présentes dans des données d'animation. Pour ce faire, nous avons présenté différentes façons d'adapter des techniques de compression par ondelettes standard à la nature des animations squelettiques. En particulier, nous avons proposé une technique d'optimisation permettant de sélectionner les coefficients d'ondelettes à conserver grâce à une recherche dans un espace réduit. Ceci nous a permis d'augmenter de façon substantielle la qualité visuelle des animations reconstruites tout en conservant des temps de décompression très rapides associés à la transformée en ondelettes. Nous avons aussi proposé une extension à cette technique permettant de corriger les artéfacts liés au glissement des pieds.

Dans les chapitres qui suivent, nous nous penchons d'abord sur la détection des redondances temporelles à grande échelle puis sur la façon de les exploiter dans le cadre de la compression de données d'animation.

Chapitre 5

Détection de sous-séquences redondantes

My superior skill has nothing to do with it ! I cannot do magic that is not there.

Mr Norrell

Dans le chapitre précédent nous nous intéressions à la mise au point d'une technique capable d'exploiter les cohérences temporelles à petite échelle qui résultent du fait que les signaux d'animation sont généralement lisses. Ces cohérences ne constituent souvent qu'une petite partie de la structure présente dans une animation et qui est susceptible d'être exploitée à des fins de compression. Par exemple, il n'est pas rare qu'une banque de données acquises par capture de mouvements contienne un grand nombre de répétitions de gestes similaires. L'identification de ces gestes peut permettre la production d'un "dictionnaire" qui, à sont tour, rend possible le développement d'une compression efficace.[1]

Nous nous attardons dans un premier temps à la génération du "dictionnaire" ; son utilisation dans le cadre de la compression et de la synthèse de mouvements sera abordée dans les prochains chapitres. Nous présentons donc une technique efficace permettant la segmentation automatique d'une banque de données d'animations en sous-séquences redondantes. Les travaux présentés ici ont été publiés dans un rapport technique [BvP07b] et sous forme de poster au *Symposium on Computer Animation 2007* [BvP07a].

[1] Dans ce chapitre, n, m, k et $\mathbf{d}$ désignent respectivement $n^{\mathtt{D}}$, $m^{\mathtt{D}}$, $k^{\mathtt{I}}$ et $\mathbf{d}^{\mathtt{I}}$. Notez que $k^{\mathtt{I}} = k^{\mathtt{D}}$, mais nous les distinguons néanmoins par souci d'uniformité.

5.1 Travaux antérieurs reliés

L'identification de sous-séquences redondantes se rapproche par plusieurs points de certains des travaux antérieurs déjà présentés au chapitre 3. Cette section résume les travaux pertinents et identifie les différences avec l'approche que nous avons adoptée.

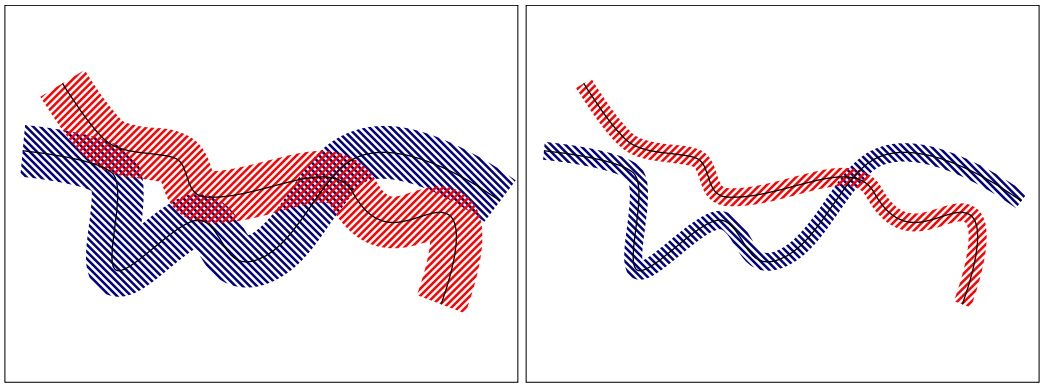
Plusieurs chercheurs ont proposé des techniques permettant de segmenter des animations acquises par capture de mouvements (section 3.3.2). À cette fin, certains ont utilisé des caractéristiques locales, comme la vitesse ou l'accélération angulaire [Bin00, Zha01, FMJ02, KPS03, KTP04, BSC05]. L'utilisation de la *PCA* ou de la *PPCA* a aussi été proposée [BSP$^+$04, LM06]. Toutes ces techniques ont la particularité d'analyser les données d'animation à l'intérieur d'une fenêtre temporelle relativement petite. À l'opposé, la segmentation que nous proposons considère l'ensemble de la banque de données.

Les techniques de segmentation par histogramme [SB05b], par agrégation de poses [BSP$^+$04, LZWM05] ou la segmentation indirecte résultant de la création d'un graphe de mouvements [AF02, KGP02, GSKJ03, SO06] sont basées sur une analyse de l'ensemble des données. Les poses sont cependant considérées isolément, ou alors seulement un voisinage restreint autour de chaque pose est utilisé. Nous proposons plutôt de segmenter les données en fonction de la présence de longues sous-séquences redondantes.

Certaines techniques de recherche de mouvements (*motion query*) [KG04b] ont aussi été utilisées pour segmenter des données dans le but de créer un graphe de mouvement paramétrique [HG07]. Au contraire de notre technique, cette approche exige un travail manuel important car l'usager doit fournir le mouvement à rechercher.

La technique que nous proposons dans la suite de ce chapitre se base sur une représentation simplifiée de la banque de données sous la forme d'une chaîne de caractères. Müller *et al.* [MRC05, MR06] ont proposé une simplification semblable dans laquelle un certain nombre de caractéristiques binaires sont utilisées pour classifier les poses. Ceci demande cependant de déterminer à l'avance l'ensemble des caractéristiques à évaluer.

La simplification proposée par Liu *et al.* [LZWM05] se rapproche beaucoup de notre représentation par chaîne de caractères. La principale différence vient du fait que nous introduisons le concept de matrice de similarité de façon à assurer la robustesse de la comparaison de sous-chaînes. Ceci nous permet de segmenter efficacement la totalité de la banque de données, alors que les signatures de Liu *et al.* sont utilisées pour la recherche d'un unique mouvement.



(a) (b)

FIG. 5.1 – Cette figure montre l'effet du critère de tolérance, représenté par l'épaisseur des traits hachurés, sur la détection de sous-séquences similaires. En (a), un critère de tolérance élevé permet d'identifier quatre paires de courtes sous-séquences similaires, indiquées par les endroits où les hachures se chevauchent. En (b), le critère de tolérance est plus faible et une seule paire de sous-séquences est identifiée. En général un critère de tolérance élevé aura tendance à générer un grand nombre de courtes sous-séquences similaires. Ce problème est encore plus important lorsque la dimensionnalité des données augmente.

5.2 Idée directrice

Bien que plusieurs des approches précédentes se soient avérées efficaces dans les contextes auxquels elles se destinent, il semble qu'aucune ne puisse être directement appliquée au problème de l'identification automatique de sous-séquences redondantes. Cette section introduit donc les concepts généraux sous-jacents à la technique que nous proposons.

En vue d'éliminer le besoin d'une segmentation préalable, notre technique s'inspire des approches par courbe d'enregistrement [KG03] utilisées dans certaines techniques de recherche de mouvements [KG04b]. Ces approches permettent d'identifier des similarités entre de très courtes sous-séquences et demandent donc l'utilisation d'un critère de tolérance très faible. En effet, le nombre de courtes sous-séquences similaires augmente rapidement avec le critère de tolérance, tel qu'illustré à la figure 5.1. En revanche, l'utilisation d'un critère de tolérance faible exige une mesure précise de la distance entre deux poses.

Notre technique relaxe de manière importante le critère de tolérance entre les poses et s'attarde plutôt à identifier des sous-séquences demeurant similaires sur une longue période de temps. La clé de notre approche, inspirée des travaux de Liu *et al.* [LZWM05], est une

représentation de la totalité de la banque de données d'animations sous la forme d'une unique chaîne de caractères. Chaque caractère identifie un ensemble de poses plus ou moins similaires et la chaîne encode les transitions entre ces groupes de poses. Ainsi, des sous-chaînes similaires permettent d'identifier des sous-séquences dont les poses progressent de la même façon, c'est-à-dire des sous-séquences redondantes.

Grâce à la grande rapidité des opérations de recherche sur les chaînes de caractères, nous pouvons développer un algorithme explorant un grand nombre de façons de partitionner la banque de données en sous-séquences redondantes. Un unique paramètre ρ, défini par l'usager, permet de favoriser tantôt les partitions contenant de longues sous-séquences, tantôt les partitions produisant des groupes qui contiennent un grand nombre de sous-séquences.

Il est important de distinguer cette transformation de la banque de données en une chaîne de caractères des techniques de recherche requérant une présegmentation [Het04]. En effet, les caractères n'indiquent pas où les sous-séquences débutent et se terminent. Certains caractères constitueront des frontières de segmentation et d'autres se trouveront en plein coeur des sous-séquences segmentées. De plus, la technique que nous proposons effectue une segmentation précise à la toute fin du processus, au cours d'une étape où les poses extrêmes des sous-séquences sont alignées.

5.3 Aperçu de la technique

Un aperçu schématique de notre technique est illustré à la figure 5.2. Chacune des étapes est abordée rapidement dans cette section et fera l'objet d'une explication détaillée dans la suite du chapitre. Pour permettre de mieux présenter la technique nous introduisons maintenant un exemple jouet, illustré à la figure 5.3. Afin d'en alléger la présentation graphique, la banque de données utilisée dans cet exemple contient un grand nombre de courtes animations ne correspondant pas à des gestes humains. De plus, ces animations n'utilisent que trois degrés de liberté. Elles sont illustrées à la figure 5.3(a) sous forme paramétrique et chaque pose y est marquée d'un $\times$ rouge. Une rupture dans la courbe paramétrique indique le début ou la fin d'une des animations contenue dans la banque de données.[2]

[2]Dans ce chapitre, toutes les figures représentant des trajectoires (*e.g.* figure 5.3(b)), des matrices de similarité (*e.g.* figure 5.3(d)), ou des chaînes de caractères (*e.g.* figure 5.3(c)) ont été générées automatiquement à l'aide de l'implantation *Matlab* de notre algorithme.

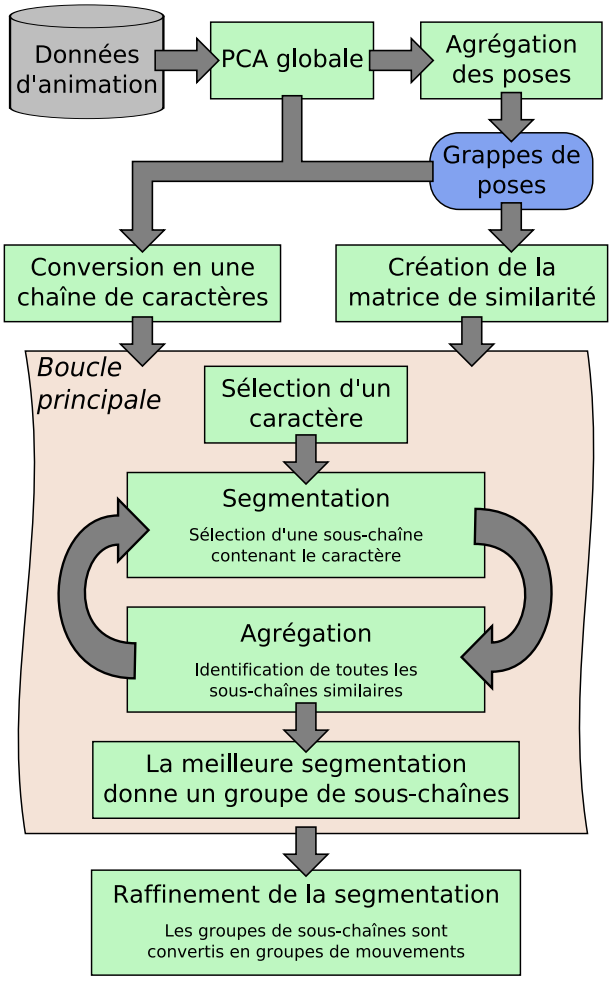
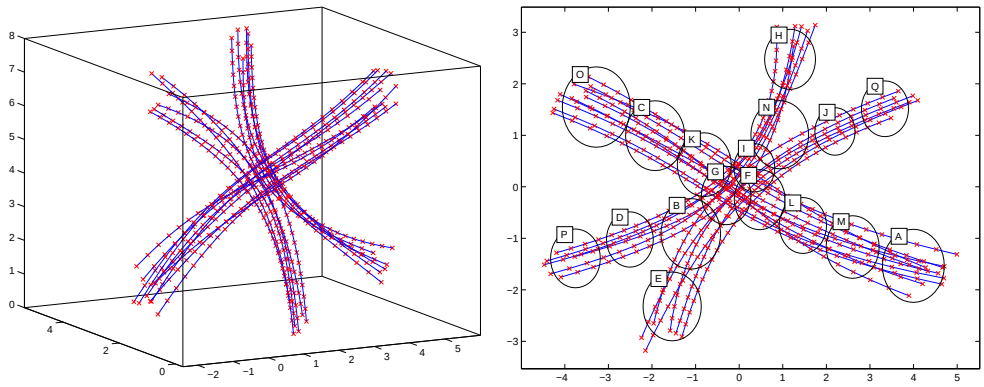


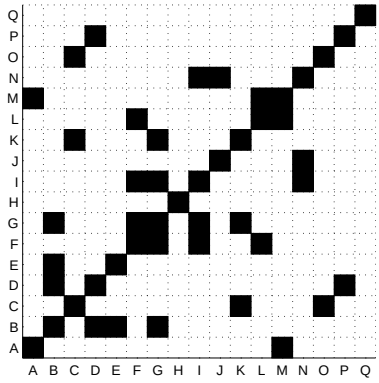
FIG. 5.2 – Aperçu schématique de la technique de partition en groupes de mouvements.



(a) Données initiales. (b) *PCA* et agrégation.

$c =$	AMLFKCOAMLFGKCOEBGINJQEBGINJQPDBGINHPDBGINHAMLFKCOPD...
$b =$	00000010000000100000010000001000000100000010000000100000010...

(c) Chaîne de caractères de la banque de données et signal binaire b.



(d) Matrice de similarité.

FIG. 5.3 – Les différentes étapes de prétraitement illustrées sur un exemple jouet.

$c =$	AMLFKCOAMLF G KCOEBGINJQEBGINJQPDBGINHPDBGINHAMLFKCOPD...
$b =$	00000010000000100000010000001000000100000010000000100000010...

(a) Choix d'une instance d'un caractère.

$c =$	AMLFKCOA MLFGKC OEBGINJQEBGINJQPDBGINHPDBGINHAMLFKCOPD...
$b =$	00000010000000100000010000001000000100000010000000100000010...

(b) Segmentation : Une sous-chaîne contenant le caractère est choisie.

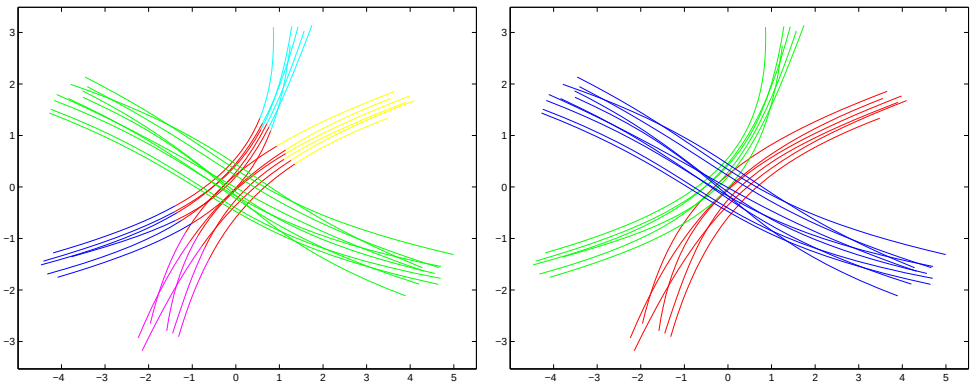
$c =$	A MLFKC OA MLFGKC OEBGINJQEBGINJQPDBGINHPDBGINHA MLFKC OPD...
$b =$	00000010000000100000010000001000000100000010000000100000010...

(c) Agrégation : Les sous-chaînes similaires sont identifiées.

$c =$	AMLFKCO AMLFGKCO EBGINJQEBGINJQPDBGINHPDBGINH AMLFKCO PD...
$b =$	1111111 11111111 00000010000000100000001000000 11111111 0...

(d) Le groupe de sous-chaînes ayant le plus grand volume est retenu.

FIG. 5.4 – Les différentes étapes de la boucle principale illustrées sur un exemple jouet.



(a) Avec $\rho = 1$. (b) Avec $\rho = 4$.

FIG. 5.5 – Les deux résultats possibles de la technique sur l'exemple jouet, en fonction de ρ. Une petite valeur de ρ favorise des groupes contenant un plus grand nombre de mouvements. Une grande valeur de ρ favorise des groupes contenant des mouvements plus longs. Chaque groupe est représenté par une couleur différente.

La première étape consiste à utiliser la *PCA* pour réduire la dimensionnalité des poses. Par la suite, ces poses sont agrégées (*clustered*) en utilisant un mélange de gaussiennes isotropes. Chacune des grappes (*clusters*) ainsi formées est étiquetée par un caractère différent, tel qu'illustré à la figure 5.3(b). La technique du maximum de vraisemblance (*maximum likelihood*) est ensuite utilisée pour placer les poses dans différentes grappes et ainsi assigner un caractère à chaque pose. On transforme finalement la séquence des poses contenues dans la banque de données en une chaîne de caractères. Cette chaîne est simplifiée en remplaçant plusieurs copies successives d'un même caractère par une seule instance de ce caractère, résultant en la chaîne présentée à la figure 5.3(c).

Conjointement à cette chaîne, on maintient un signal binaire b contenant une entrée de moins que le nombre de caractères dans la chaîne. C'est-à-dire qu'une valeur binaire est associée à chaque intervalle entre deux caractères. Le signal b est entièrement initialisé à 0, sauf pour les intervalles entre deux caractères appartenant à des animations différentes, où il est initialisé à 1. Par la suite, un caractère sera dit *segmenté* si les deux valeurs de b correspondant à l'intervalle précédent et suivant sont égales à 1.

Comme on l'a expliqué précédemment, on considère que les poses contenues dans une même grappe sont toutes similaires. Ceci a cependant l'inconvénient d'introduire des effets de bord à l'interface entre deux grappes. En effet, il est possible qu'en ces endroits deux poses très similaires se voient assignées à des grappes différentes. Pour remédier à ce problème, nous introduisons une matrice binaire de similarité indiquant quelles grappes sont assez rapprochées pour contenir des poses similaires. Pour notre exemple jouet, cette matrice est présentée à la figure 5.3(d).

Les étapes précédentes constituent l'initialisation de l'algorithme. Par la suite, la boucle principale, illustrée à la figure 5.4, s'occupe de segmenter la banque de données et de grouper les segments obtenus en sous-séquences similaires. Il est important de remarquer que la segmentation et l'agrégation des sous-séquences sont effectuées en un processus coordonné. Il s'agit d'une originalité importante de notre approche, rendue possible par l'efficacité des algorithmes de manipulation des chaînes de caractères.

La première étape de la boucle principale consiste à identifier le caractère le plus fréquemment rencontré dans la chaîne. Une instance de ce caractère est choisie aléatoirement et constitue le point de départ de la boucle de segmentation-agrégation. Dans notre exemple, une instance du caractère "G" est choisie à la figure 5.4(a).

La boucle de segmentation-agrégation s'intéresse successivement à toutes les sous-chaînes non segmentées contenant l'instance de caractère choisie, et ce jusqu'à une longueur de sous-chaîne maximale λ spécifiée par l'usager. Pour chaque sous-chaîne ainsi obtenue, l'algorithme identifie toutes les sous-chaînes similaires en prenant en considération la matrice de similarité. Dans notre exemple, à la figure 5.4(b), la sous-chaîne "MLFGKC" est présentement explorée. On voit ensuite à la figure 5.4(c) que deux autres sous-chaînes similaires de la forme "MLFKC" ont été identifiées.

Le résultat est un ensemble que nous nommons un groupe de sous-chaînes. Pour évaluer la qualité d'un groupe de sous-chaînes nous introduisons le concept de volume. Plus un groupe contient de sous-chaînes, plus son volume est important. De la même façon, le volume d'un groupe augmente avec la longueur des sous-chaînes qu'il contient.

La boucle de segmentation-agrégation génère ainsi un grand nombre de groupes potentiels de sous-chaînes. Parmi ceux-ci, seul celui ayant le plus grand volume est retenu. Les sous-chaînes contenues dans ce groupe sont segmentées en plaçant une suite de 1 dans le signal b aux positions appropriées, tel qu'illustré à la figure 5.4(d). La boucle principale est ensuite exécutée à nouveau, et ce jusqu'à ce que b ne contienne que des 1.

La dernière étape du processus consiste en un raffinement de la segmentation. Ainsi, pour chaque sous-chaîne segmentée, on identifie une pose de début et une pose de fin. Ces poses sont choisies de façon à minimiser la distance avec toutes les poses de début ou de fin contenues dans un même groupe de sous-chaînes. Ce faisant, chaque sous-chaîne est transformée en une sous-séquence identifiée par un intervalle entre deux poses.

Le résultat de l'algorithme est donc un ensemble de groupes de mouvements. Plusieurs résultats différents peuvent être intéressants, dépendamment que l'on privilégie les mouvements plus longs ou les groupes plus nombreux. Pour refléter cela, un paramètre ρ spécifié par l'usager permet d'influencer la calcul du volume. Celui-ci permet de favoriser les groupes possédant beaucoup de sous-chaînes ou plutôt ceux contenant des sous-chaînes plus longues. Dans l'exemple jouet, ceci peut se traduire par l'obtention d'une des deux solutions illustrées à la figure 5.5.

5.4 Prétraitement

L'aperçu présenté précédemment décrit brièvement toutes les étapes de la technique de création de groupes de mouvements. Si cet aperçu est suffisant pour saisir l'essentiel de notre

approche, il n'en demeure pas moins succinct et omet bon nombre de subtilités. Les sections qui suivent visent à explorer en profondeur les différentes étapes de l'algorithme. Nous y abordons entre autres les détails d'implantation, les différents paramètres accessibles à l'utilisateur, la sensibilité de la technique à ces paramètres ainsi que la justification et l'importance des différentes étapes. Pour débuter, nous décrivons les opérations de prétraitement effectuées avant l'exécution du corps de l'algorithme.

5.4.1 Représentation des poses

Il existe un grand nombre de manières différentes de représenter les poses présentes dans une banque de données. Comme nous l'avons déjà mentionné à la section 2.2.1, les données brutes que nous possédons représentent les poses à l'aide de la position de la racine et de la valeur angulaire de chacun des autres degrés de liberté. L'algorithme que nous décrivons pourrait cependant fonctionner avec n'importe quelle autre représentation des poses. On pourrait par exemple utiliser les positions de chaque joint, ou encore la représentation par nuage de points introduite par Kovar et Gleicher [KG04b].

Pour bien choisir l'espace de représentation des poses, il importe de comprendre qu'un des objectifs de l'algorithme consiste à identifier des sous-séquences similaires. On souhaite donc une représentation dans laquelle deux poses considérées similaires seront encodées par deux vecteurs rapprochés. Par exemple, l'angle azimutal de la racine ne semble pas approprié étant donné que nous considérons que deux mouvements sont similaires même si le personnage ne fait pas face à la même direction. Il en va de même pour la position de la racine dans le plan puisque deux mouvements sont similaires peu importe la position d'où ils originent.

D'autre part, on sait que la performance et l'efficacité de l'agrégation (*clustering*) des poses diminue lorsque la dimensionnalité de l'espace augmente. On souhaite donc la représentation la plus compacte possible qui conserve néanmoins toutes les caractéristiques des poses. Pour ce faire, il est préférable de représenter les poses à l'aide de vecteurs où les dimensions sont le plus indépendantes possible.

Ainsi, bien que les représentations à base de positions ou de nuages de points soient plus aptes à bien saisir les similarités et les différences entre les poses, elles le font au détriment d'une augmentation importante du nombre de dimensions. C'est pourquoi nous préférons une représentation basée plutôt sur les angles. Nous pouvons nous permettre de choisir une telle représentation même si elle évalue moins bien la distance entre deux poses précisément parce

que, tel qu'indiqué à la section 5.2, nous utilisons un critère de tolérance moins précis que celui de Kovar et Gleicher [KG04b].

On sait que, dans notre représentation, un squelette à k degrés de liberté possède $k-6$ degrés de liberté angulaires non attachés à la racine : $\mathbf{d}_7^{\mathtt{D}}, ..., \mathbf{d}_k^{\mathtt{D}}$. À ceux-ci nous ajoutons les angles d'élévation et de roulement de la racine : $\mathbf{d}_4^{\mathtt{D}}, \mathbf{d}_6^{\mathtt{D}}$. Pour construire le vecteur d'une pose, chacun de ces angles est exprimé en radians et multiplié par une pondération $\mathbf{w}_i$ égale à la longueur de la plus longue chaîne d'os reliant le degré de liberté concerné à une feuille du squelette (voir section 2.3.2). Cette pondération a deux avantages : elle transforme les degrés de liberté angulaires de façon à les rendre comparables aux positions qui seront ajoutées plus tard, et elle permet de prendre en considération l'importance relative de chacun des joints. Cette dernière caractéristique fait en sorte que l'espace construit offre une meilleure représentation de la similarité entre les poses que l'espace angulaire direct.

Pour compléter cette représentation, nous utilisons le fait que tous les mouvements de notre banque de données se produisent sur un plan de hauteur $y = 0$. Ainsi, la position verticale de la racine $\mathbf{d}_2^{\mathtt{D}}$ constitue une caractéristique permettant de distinguer deux poses différentes et nous l'ajoutons à notre représentation.[3]

Pour terminer, nous remarquons que nous avons laissé de côté la position horizontale et l'angle azimutal de la racine. Comme nous l'avons expliqué plus haut, ceci est dû au fait que nous considérons que deux poses sont similaires peu importe leur position et leur orientation dans le plan. Il n'en demeure pas moins qu'un mouvement de marche tournant vers la gauche est différent d'un mouvement de marche tournant vers la droite. Cependant, avec notre représentation actuelle, les poses de ces mouvements peuvent être en tout point identiques. Pour remédier à ce problème, nous ajoutons à la représentation les vitesses horizontales $\dot{\mathbf{d}}_1^{\mathtt{D}}, \dot{\mathbf{d}}_3^{\mathtt{D}}$ et la vitesse angulaire azimutale $\dot{\mathbf{d}}_5^{\mathtt{D}}$.

La représentation finale que nous utilisons pour une pose est donc

$$\mathbf{d} = \left(\dot{\mathbf{d}}_1^{\mathtt{D}}, \mathbf{d}_2^{\mathtt{D}}, \dot{\mathbf{d}}_3^{\mathtt{D}}, \mathbf{w}_4\mathbf{d}_4^{\mathtt{D}}, \mathbf{w}_5\dot{\mathbf{d}}_5^{\mathtt{D}}, \mathbf{w}_6\mathbf{d}_6^{\mathtt{D}}, \mathbf{w}_7\mathbf{d}_7^{\mathtt{D}}, ..., \mathbf{w}_k\mathbf{d}_k^{\mathtt{D}}\right) .$$

La totalité d'une banque de données d'animations comptant n échantillons peut donc s'écrire sous la forme d'une séquence $\mathbf{d}(i), 1 \leq i \leq n$. De plus on introduit la notation $\mathbf{d}[i,j]$ pour indiquer la sous-séquence $\mathbf{d}(i), \mathbf{d}(i+1), ..., \mathbf{d}(j)$.

[3] Si les données comprennent des mouvements se déroulant sur des plans de différentes hauteurs, par exemple un acteur montant un escalier, il est possible de remplacer $\mathbf{d}_2^{\mathtt{D}}$ par la vitesse verticale de la racine, $\dot{\mathbf{d}}_2^{\mathtt{D}}$.

5.4.2 Projection des poses

La représentation vectorielle choisie est compacte car chaque dimension représente des degrés de liberté différents qui sont *a priori* indépendants. Cependant, on sait que la physiologie du mouvement humain fait en sorte que ces degrés de liberté sont souvent coordonnés. Pour réduire de façon encore plus importante la dimensionnalité, nous projetons l'ensemble des données (vecteurs représentant les différentes poses au cours du temps) sur une base plus petite obtenue grâce à la *PCA*.

Nous avons déjà introduit la *PCA* et détaillé son fonctionnement à la section 4.8.4. Cependant, à la différence de la compression, nous ne souhaitons pas conserver un nombre précis de dimensions, mais plutôt effectuer une projection qui maintienne un certain critère de qualité. Pour ce faire, le nombre de dimensions est sélectionné de façon à conserver un pourcentage d'énergie α spécifié par l'usager. Suivant les définitions usuelles en analyse des composantes principales, l'énergie après projection s'obtient en sommant les valeurs propres correspondant aux vecteurs retenus. L'énergie totale est simplement la somme de toutes les valeurs propres obtenues lors de l'application de la *PCA*.

En pratique, nous avons utilisé une valeur $\alpha = 70\%$ pour tous les résultats présentés plus bas. Ceci résulte en la conservation de 7 à 12 dimensions sur les 62 originales. Le nombre de dimensions conservé est noté $k^{\texttt{PCA}}$. Nous avons remarqué que l'application de la *PCA* n'améliore ni ne détériore la qualité des résultats obtenus. Cependant, cette opération de projection globale permet de réduire de beaucoup le temps consacré à l'agrégation des poses, qui constitue actuellement le principal goulot d'étranglement de notre technique.

5.4.3 Agrégation des poses

La création des grappes de poses s'effectue dans un premier temps en modélisant les données à l'aide d'une distribution par mélange de gaussiennes isotropes. Par isotropes, on entend qu'une gaussienne $\mathcal{G}$ a une moyenne vectorielle $\boldsymbol{\mu} \in \mathbb{R}^{k^{\texttt{PCA}}}$ et une matrice de covariance proportionnelle à l'unité, $\boldsymbol{\Sigma} = \sigma\mathbf{I}$. Chaque composante du mélange compte donc $k^{\texttt{PCA}} + 1$ paramètres.

Nous avons expérimenté différents modèles de distributions et avons remarqué que les meilleurs résultats étaient obtenus lorsque le modèle possède un grand nombre de composantes, chacune munie d'un nombre assez restreint de paramètres. C'est pour cette raison que nous

avons préféré des distributions isotropes plutôt que des gaussiennes possédant une matrice de covariance diagonale ou quelconque.

Le seul paramètre spécifié par l'usager pour cette étape est κ, le nombre de gaussiennes désirées dans la distribution. Il est important de remarquer que κ indique aussi le nombre de grappes qui seront générées et donc le nombre de caractères différents qui composeront la chaîne de caractères. De fait, on étiquette chaque gaussienne $\mathcal{G}_i$ du modèle par un caractère i différent.

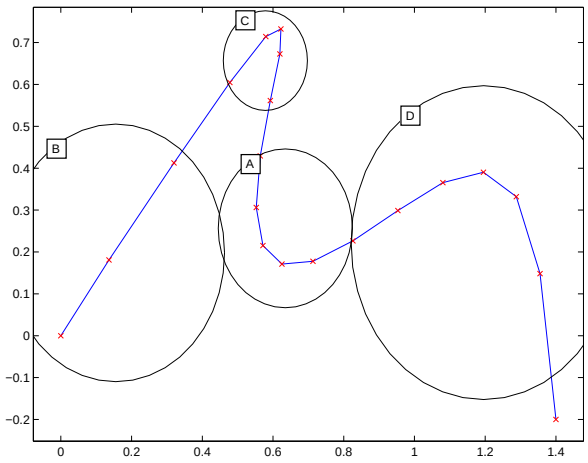
Nous utilisons la technique standard d'*Expectation Maximization* (*EM*) pour identifier les paramètres de gaussiennes maximisant la vraisemblance des données [Bis95]. Une fois ces paramètres connus, on peut procéder à l'agrégation des poses de la banque de données en utilisant la technique du maximum de vraisemblance pour associer une grappe à chaque pose.

5.4.4 Conversion en une chaîne de caractères

L'agrégation des poses nous permet de convertir la banque de données de n poses en une chaîne de n caractères, où le caractère associé à une pose correspond à l'étiquette de la grappe contenant cette pose. Cette chaîne contient de fréquentes répétitions successives d'un même caractère. Le nombre de répétitions d'un caractère permet de connaître la vitesse à laquelle l'animation traverse la grappe correspondante. Cette information n'est cependant pas nécessaire étant donné que, dans l'esprit des techniques de *dynamic time-warping* [BW95], on souhaite considérer que deux mouvements sont similaires même s'ils ne s'exécutent pas à la même vitesse.

On procède donc à une simplification de la chaîne en remplaçant toute séquence de caractères répétés par une seule instance de ce caractère. Ceci produit une chaîne de m caractères : $c(1), ..., c(m)$. Cette simplification fait qu'à chacun des caractères $c(i)$ correspond un certain nombre de poses de la banque de données originale. On note $F(i)$ l'ensemble des indices des poses représentées par le caractère $c(i)$ (figure 5.6). On note aussi $\bar{F}(i)$ la moyenne des indices contenus dans $F(i)$, arrondie à l'entier le plus proche. Finalement, on introduit la notation $c[i, j]$ pour identifier la sous-chaîne $c(i), ..., c(j)$.

Pour terminer, notons que notre définition d'une "chaîne de caractères" s'éloigne un peu de la définition usuelle en informatique. En effet, nous ne nous limitons pas ici aux séquences de codes *ASCII* où chaque caractère est encodé sur 8 bits. Notre définition admet en effet des alphabets arbitrairement grands. Dans ce contexte, une chaîne de m caractères sur un alphabet



(a) Grappes de poses.

BBBCCCCCAAAAADDDDDDD

(b) Chaîne de caractères avant simplification.

$c =$	BCAD
$\bar{F} =$	$2, 6, 11, 17$

(c) Chaîne de caractères simplifiée.

FIG. 5.6 – Exemple de conversion en une chaîne de caractères. (a) Les poses sont agrégées, puis (b) assignées à une grappe par maximum de vraisemblance, et finalement (c) les répétitions successives sont éliminées, les ensembles F et leurs moyennes $\bar{F}$ sont calculés. Dans cet exemple $F(1) = \{1, 2, 3\}$, $F(2) = \{4, 5, 6, 7, 8\}$, etc.

de κ éléments n'est ni plus ni moins qu'un vecteur d'entiers dans $\{1, ..., \kappa\}^m$. Lorsque κ est assez petit on peut faire correspondre un caractère imprimable (*i.e.* un caractère) à chaque indice, comme nous l'avons fait à la figure 5.6. Le terme "chaîne de caractères" a été retenu parce qu'il nous semble favoriser une compréhension intuitive de la technique en plus d'illustrer la nature des algorithmes de recherche qui seront introduits plus loin.

5.4.5 Initialisation du signal de partitionnement

En plus de la chaîne de caractères de longueur m encodée dans le signal c, on maintient un signal binaire b de taille $m - 1$. Chaque élément de ce signal de partitionnement correspond à un intervalle entre deux caractères du signal c. Une valeur de $b(i) = 0$ indique que l'intervalle entre les caractèress $c(i)$ et $c(i + 1)$ n'a pas encore été utilisé dans un groupe de sous-chaînes.

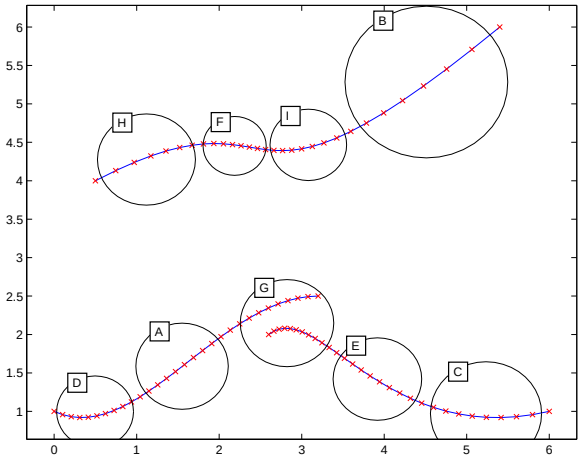
Ainsi, b est normalement entièrement initialisé à 0. Il est cependant possible que la banque de données comprenne plusieurs animations distinctes. La transition de la dernière pose d'une animation à la première pose de l'animation suivante est une rupture brutale qui ne correspond pas à une transition valide. En supposant que les caractères correspondant à ces deux poses soient respectivement $c(i)$ et $c(i + 1)$, alors on initialise $b(i) = 1$. Si les deux poses sont représentées par un même caractère, alors on dédouble préalablement ce caractère. Ces cas sont illustrés à la figure 5.7.

Les poses correspondant à des changements d'animations sont généralement spécifiées dans le fichier d'entrée, mais elles peuvent aussi être déduites facilement de façon automatique en analysant les discontinuités dans le vecteur des poses. C'est cette dernière approche que nous avons utilisée.

5.4.6 Matrice de similarité des grappes

Comme nous l'avons vu lors de l'aperçu de la technique, il est nécessaire de pouvoir déterminer si deux caractères distincts sont similaires ou non. Pour ce faire, nous construisons une matrice de similarité binaire $\mathbf{A}$ telle qu'un élément $\mathbf{A}_{i,j}$ est égal à 1 si le caractère i est similaire au caractère j, et 0 sinon.

Pour construire cette matrice, nous nous intéressons aux distances entre les gaussiennes associées à chaque caractère. Ces distances pourraient être définies à partir de la distance euclidienne entre les moyennes. Cependant, cette définition ne prend pas en considération la variance des gaussiennes. Il apparaît en effet logique que deux gaussiennes très localisées (*i.e.* à faible

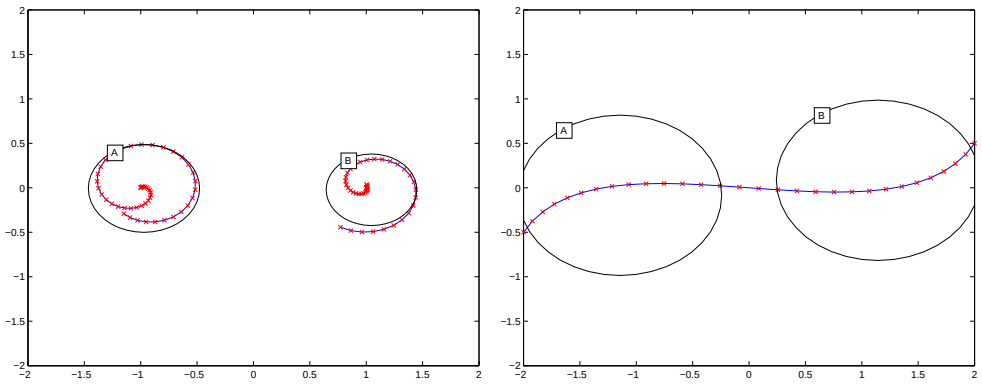


(a) Grappes de poses.

$c =$	HFIBDAGGEC
$b =$	000100100

(b) Chaîne de caractères et signal b.

FIG. 5.7 – Exemple illustrant l'initialisation du signal binaire b en présence de trois animations distinctes. Le caractère "G" a été dédoublé car il apparaît au début et à la fin de deux animations consécutives.



(a) (b)

FIG. 5.8 – Deux exemples sur lesquels on a procédé à l'agrégation des poses en deux grappes. Chaque gaussienne en (a) a approximativement la même moyenne que la gaussienne correspondante en (b). Les variances sont cependant bien différentes : en (a) elles sont faibles et on souhaite considérer les deux grappes comme éloignées, en (b) elles sont plus grandes et on souhaite considérer les deux grappes comme plus rapprochées.

variance) soient considérées plus éloignées que deux gaussiennes de mêmes moyennes mais à variances plus élevées (figure 5.8). Pour pallier à ce problème, nous utilisons la distance de Mahalanobis [Mah36], qui vise précisément à prendre la variance en considération. Pour une gaussienne $\mathcal{G}_i$ de moyenne $\boldsymbol{\mu}_i$ et de covariance $\sigma_i \mathbf{I}$, la distance de Mahalanobis est définie par

$$\mathcal{D}(\mathcal{G}_i, \mathbf{v}) = \frac{|\mathbf{v} - \boldsymbol{\mu}_i|}{\sigma_i} \ . \tag{5.1}$$

On définit donc la distance entre deux gaussiennes par le maximum des distances de Mahalanobis entre les moyennes,

$$\mathcal{D}(\mathcal{G}_i, \mathcal{G}_j) = \max(\mathcal{D}(\mathcal{G}_i, \boldsymbol{\mu}_j), \mathcal{D}(\mathcal{G}_j, \boldsymbol{\mu}_i)) \ .$$

Cette expression peut être simplifiée de la façon suivante,

$$\mathcal{D}(\mathcal{G}_i, \mathcal{G}_j) = \left| \frac{\boldsymbol{\mu}_i - \boldsymbol{\mu}_j}{\min(\sigma_i, \sigma_j)} \right| \ .$$

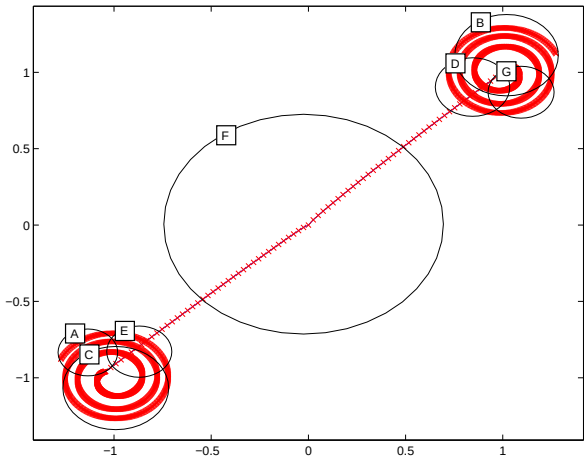
La matrice $\mathbf{A}$ est obtenue simplement en calculant les distances entre toutes les paires de grappes et en utilisant un seuil τ au-delà duquel les grappes sont considérées trop éloignées :

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{si } \mathcal{D}(\mathcal{G}_i, \mathcal{G}_j) \leq \tau \ , \\ 0 & \text{sinon.} \end{cases}$$

En général, on peut considérer τ comme un paramètre de la technique qui doit être spécifié par l'usager. Ce paramètre est cependant difficile à déterminer étant donné qu'il dépend à la fois de la base retenue lors de la *PCA*, du nombre de gaussiennes utilisées pour l'agrégation et de la nature des animations présentes dans la banque de données. Pour contourner ce problème nous introduisons maintenant une technique qui vise à déterminer automatiquement une bonne valeur pour τ.

Nous remarquons dans un premier temps que, pour éviter les effets de bord mentionnés à la section 5.3, deux grappes contenant des poses voisines dans l'animation devraient être considérées similaires. Autrement dit, deux caractères $c(i)$ et $c(i+1)$ apparaissant côte-à-côte dans la chaîne de caractères devraient être considérées similaires sauf si $b(i) = 1$ indiquant que ces caractères correspondent à des animations différentes.

Cette observation nous mène vers une première façon simple de déterminer τ : il suffit simplement de le fixer à la distance maximum observée entre deux caractères adjacents dans la chaîne. Cette approche est cependant très sensible aux observations aberrantes (*outliers*) qui peuvent survenir. Par exemple, certaines régions de l'espace des poses contiennent parfois peu d'échantillons distants les uns des autres. Ces régions ne seront généralement pas bien



(a) Grappes de poses.

$c =$	...CAECAEFDBGDBG...

(b) Chaîne de caractères.

FIG. 5.9 – Un exemple où la distance entre les grappes correspondant à deux caractères consécutifs est en général assez faible sauf entre “E” et “F” et entre “F” et “D”. De telles observations aberrantes se rencontrent assez fréquemment en pratique.

modélisées par les gaussiennes, faisant en sorte que deux caractères adjacents correspondent à des grappes assez distantes l'une de l'autre. Ce problème est illustré à la figure 5.9.

Nous préférons donc utiliser une évaluation basée sur les quantiles et plus robuste aux observations aberrantes. On exprime donc τ de la façon suivante,

$$\tau = \text{quant}\left(\left\{\mathcal{D}(\mathcal{G}_{c(i)}, \mathcal{G}_{c(i+1)}) \;\middle|\; 1 \leq i < m \;\text{ et }\; b(i) = 0\right\}, \tau'\right) \;,$$

où τ' est un quantile spécifié par l'usager, généralement près de 1. L'ensemble exprimé ici correspond à toutes les distances entre des caractères adjacents appartenant à une même animation. En pratique, la valeur de τ' n'a pas à être ajustée en fonction des données ou des autres paramètres de la technique. Pour tous les résultats présentés ci-après, nous avons utilisé $\tau' = 90\%$.

5.5 Boucle principale

Le coeur de l'algorithme consiste à utiliser les informations calculées lors du prétraitement pour extraire ce que nous appelons des groupes de sous-chaînes (*sub-string bundles*). Un groupe

de sous-chaînes C est un ensemble de la forme

$$C = \{c[i_1, j_1], c[i_2, j_2], ...\} \ ,$$

contenant des sous-chaînes qui sont toutes similaires. La longueur de la o-ième sous-chaîne est notée $m_o^{\mathrm{s}} = j_o - i_o + 1$.

Le nombre d'éléments dans le groupe de sous-chaînes, $|C|$, est appelé la hauteur du groupe de sous-chaînes. Nous définissons aussi la largeur du groupe $\tilde{C}$ comme étant la longueur moyenne des mouvements correspondants aux sous-chaînes contenues dans le groupe soit,

$$\tilde{C} = \frac{1}{|C|} \sum_{o=1}^{|C|} \bar{F}(j_o) - \bar{F}(i_o) + 1 \ .$$

5.5.1 Sous-chaînes similaires

Il importe dans un premier temps de bien définir ce que nous entendons par "sous-chaînes similaires". Pour ce faire, nous décrivons une relation $\mathcal{M}(i_1, j_1; i_2, j_2)$ qui sera vraie si et seulement si les sous-chaînes $c[i_1, j_1]$ et $c[i_2, j_2]$ sont similaires.

Deux propriétés désirées pour $\mathcal{M}$ sont la réflexivité et la symétrie. La première exigence vient du fait qu'une sous-chaîne doit être considérée similaire à elle-même. La seconde propriété, quant à elle, naît du fait que, si la sous-chaîne $c[i_1, j_1]$ est similaire à $c[i_2, j_2]$, la réciproque doit aussi être vraie.

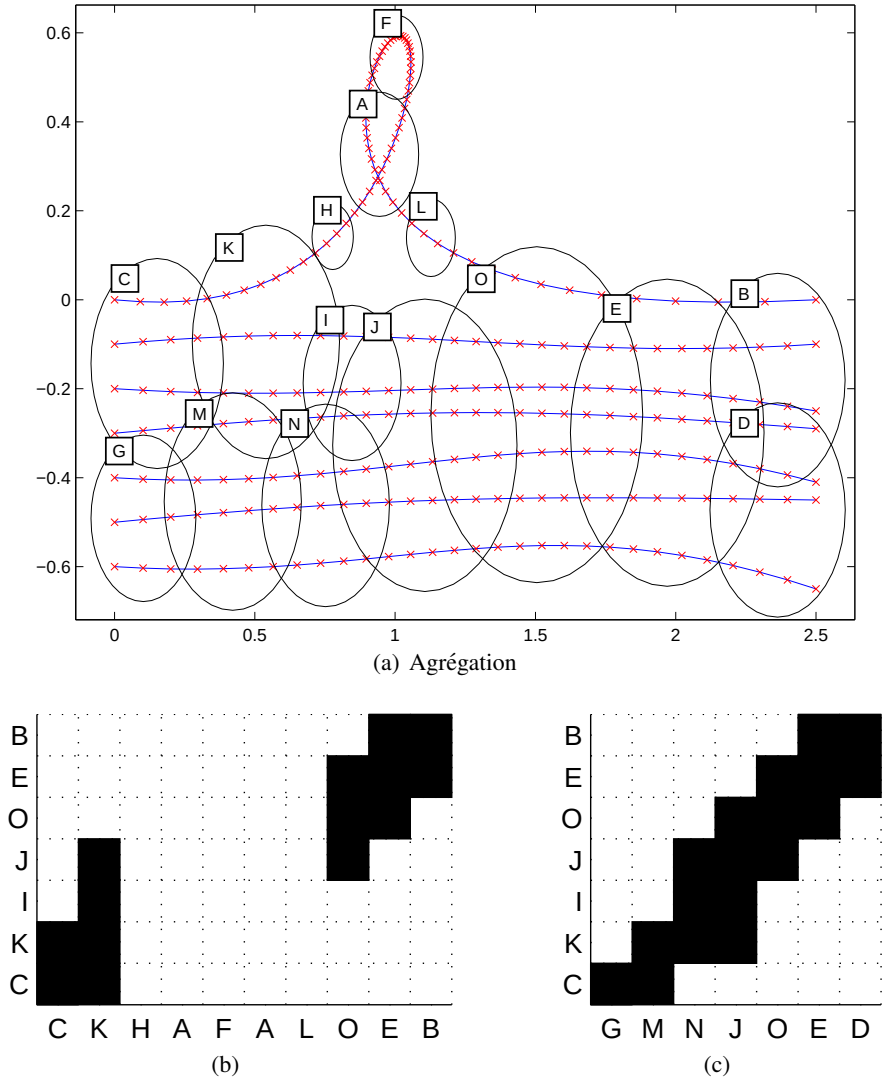
Pour construire $\mathcal{M}$, nous identifions tout d'abord quels caractères des sous-chaînes sont similaires. Pour ce faire, nous construisons une matrice de similarité entre chaque paire de poses des sous-chaînes. Formellement, cette matrice $\mathbf{S}$ est définie de la façon suivante,

$$\mathbf{S}_{o_1,o_2} = \mathbf{A}_{c(o_1+i_1-1),c(o_2+i_2-1)} \ .$$

Cette matrice est de taille $m_1^{\mathrm{s}} \times m_2^{\mathrm{s}}$ avec m_1^{s} et m_2^{s} représentant la longueur des sous-chaînes. La matrice $\mathbf{S}$ contient un 1 à chaque position où les sous-chaînes possèdent des caractères similaires et 0 ailleurs. Un exemple d'une telle matrice est donné à la figure 5.10.

Les deux sous-chaînes sont similaires seulement s'il est possible de les balayer simultanément en identifiant à tout moment une paire de caractères similaires. Autrement dit, pour que la relation $\mathcal{M}$ soit vraie, il doit exister un chemin de 1 débutant à la cellule $\mathbf{S}_{1,1}$ et se terminant à $\mathbf{S}_{m_1^{\mathrm{s}},m_2^{\mathrm{s}}}$.

L'existence de ce chemin n'est cependant pas suffisante, il faut s'assurer que tous les caractères des sous-chaînes sont visités dans le bon ordre. Pour formaliser ce concept, supposons que nous disposons d'un chemin traversant les cellules $(i_1, j_1), (i_2, j_2), ..., (i_a, j_a)$, avec



(a) Agrégation

(b) (c)

FIG. 5.10 – Cet exemple montre les matrices de similarité entre deux sous-chaînes. La figure (a) montre les données de test utilisées ainsi que les grappes résultant de l'agrégation des poses. La figure (b) montre la matrice $\mathbf{S}$ construite pour comparer les sous-chaînes "CKIJOEB" et "CKHAFALOEB". On voit qu'il n'existe aucun chemin du coin inférieur gauche au coin supérieur droit, donc les sous-chaînes ne sont pas similaires. La figure (c) montre quant à elle que les sous-chaînes "CKIJOEB" et "GMNJOED" sont similaires.

$(i_1, j_1) = (1, 1)$ et $(i_a, j_a) = (m_1^{\mathsf{S}}, m_2^{\mathsf{S}})$ et où $\mathbf{S}_{i_o, j_o} = 1, \forall o$. Pour que ce chemin soit valide il faut que chaque pas progresse d'un caractère dans au moins une des deux sous-chaînes. Plus formellement, le chemin est valide si et seulement si, pour $1 \leq o < a$,

$$\begin{aligned} &(i_{o+1}, j_{o+1}) \neq (i_o, j_o) \text{ et} \\ &i_o \leq i_{o+1} \leq i_o + 1 \text{ et} \\ &j_o \leq j_{o+1} \leq j_o + 1 \ . \end{aligned}$$

Cette définition nous donne les deux propriétés désirées pour la relation. La réflexivité de $\mathcal{M}$ vient du fait que tous les éléments sur la diagonale de $\mathbf{A}$ sont égaux à 1. La symétrie découle directement de la symétrie de la définition.

Supposons que nous disposions d'une sous-chaîne d'origine $c[i, j]$, on peut construire l'ensemble de toutes les sous-chaînes $c[i', j']$ pour lesquelles la relation $\mathcal{M}(i, j; i', j')$ est vraie. Ceci n'est cependant pas un ensemble d'équivalence étant donné que la transitivité de $\mathcal{M}$ n'est pas garantie. Néanmoins, dans le contexte de l'algorithme vorace qui sera introduit plus tard, il serait intéressant de s'assurer que nous sommes bien en présence d'un ensemble d'équivalence. En effet, ceci offrirait l'avantage de rendre le contenu d'un groupe de sous-chaînes indépendant de la sous-chaîne d'origine utilisée pour le générer, ce qui n'est pas le cas lorsque la relation $\mathcal{M}$ est utilisée directement (figure 5.11).

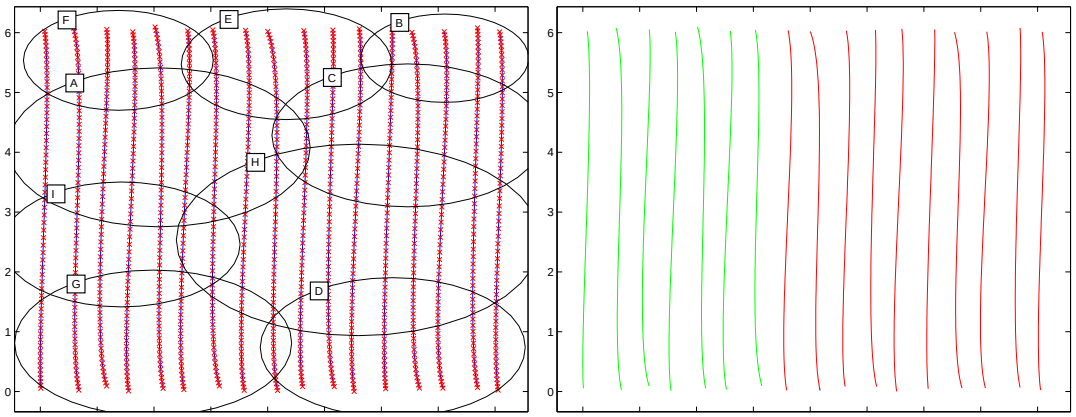
Pour produire un ensemble d'équivalence il suffit simplement d'appliquer la fermeture transitive de la relation $\mathcal{M}$. Le groupe de sous-chaînes C obtenu avec la sous-chaîne d'origine $c[i, j]$ peut donc être défini de la façon suivante,

$$C = \{c[i, j]\} \cup \left\{ c[i', j'] \ \middle| \ \exists i'', j'' \text{ t.q. } c[i'', j''] \in C \text{ et } \mathcal{M}(i', j'; i'', j'') \right\} \ .$$

En terminant, remarquons que le groupe C ne doit pas contenir de sous-chaînes se chevauchant. Pour ce faire, on le construit de manière itérative, en s'assurant que chaque nouvelle sous-chaîne qu'on y ajoute ne chevauche pas celles qui s'y trouvent déjà. Cette particularité fait en sorte qu'il existe plusieurs ensembles C pouvant être obtenus à partir d'une sous-chaîne initiale donnée. En pratique ceci peut se produire de deux façons : lorsqu'on est en présence de mouvements cycliques ou lorsqu'il existe plusieurs choix pour le premier et le dernier caractère d'une sous-chaîne similaire.

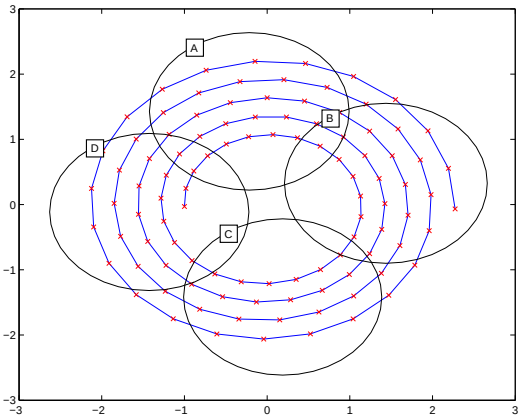
Le premier type de problème est illustré à la figure 5.12, où la sous-chaîne initiale $c[1, 7] =$ "DABCDAB" est similaire à $c[1, 7]$, $c[5, 11]$, $c[9, 15]$ et $c[13, 19]$. Comme on souhaite éviter le chevauchement des sous-chaînes, il existe trois groupes possibles,

$$\begin{aligned} &C = \{c[1, 7], c[9, 15]\} \qquad \text{ou} \\ &C = \{c[1, 7], c[13, 19]\} \qquad \text{ou} \\ &C = \{c[5, 11], c[13, 19]\} \ . \end{aligned}$$



(a) Agrégation (b) Groupes de mouvements

FIG. 5.11 – Exemple montrant plusieurs mouvements similaires qui devraient former un seul groupe de sous-chaînes. En (a) les poses sont agrégées. Les trois sous-chaînes “FAIG”, “ECHD” et “BCHD” sont similaires. En (b), “BCHD” est choisie comme sous-chaîne d’origine et la fermeture transitive n’est pas appliquée. Le groupe de sous-chaînes associé (en rouge) ne contient pas les sous-chaînes “FAIG” car “B” n’est pas similaire à “F”. L’application de la fermeture transitive corrige ce problème.



(a) Agrégation

$c =$	DABCDABCDABCDABCDAB

(b) Chaîne de caractères.

FIG. 5.12 – Exemple illustrant la construction de C en présence d’un mouvement cyclique.

La technique que nous utilisons choisira arbitrairement un de ces ensembles. En pratique, cependant, ceci ne constitue pas un réel problème étant donné que l'algorithme que nous introduirons plus bas ne retient que la sous-chaîne d'origine produisant le meilleur groupe de sous-chaînes possible. Dans cet exemple, il serait obtenu avec $c[1,4] =$ "DABC" produisant le groupe

$$C = \{c[1,4], c[5,8], c[9,12], c[13,16]\} .$$

Le second type de problème est quant à lui bien réel. En effet, la matrice de similarité fait qu'il existe souvent plusieurs choix différents pour la position de début et de fin d'une sous-chaîne similaire. Pour pallier à ce problème, on sélectionne la sous-chaîne dont les premier et dernier caractères sont les plus rapprochés de la sous-chaîne d'origine en regard de la distance entre grappes définie à l'équation 5.1. Par exemple, dans la figure 5.11, la sous-chaîne "AIG" est similaire à "ECHD", "CHD", "ECH" et "CH". Comme toutes ces sous-chaînes se chevauchent, une seule peut être incluse dans le groupe de sous-chaînes. On sélectionne "CHD" car son premier caractère et son dernier caractère sont les plus rapprochés de ceux de la sous-chaîne "GIA".

5.5.2 Choix d'un caractère initial

La première étape de la boucle principale consiste simplement à choisir une position initiale a dans la chaîne de caractères. Cette position sera utilisée par la suite pour construire et évaluer un grand nombre de groupes de sous-chaînes différents.

Le choix de a constitue l'étape vorace (*greedy step*). Ceci vient du fait qu'une fois cette position choisie, l'algorithme se commet à construire un groupe de sous-chaînes qui incluera le caractère $c(a)$. Étant donné qu'un intervalle entre deux caractères ne peut être inclus que dans un seul groupe de sous-chaînes, il est essentiel de choisir a de façon à tenter d'utiliser ces intervalles de façon optimale.

De façon à mieux diriger le choix de la position initiale, on choisit a de façon à ce que le caractère $c(a)$ corresponde à celui dont la fréquence est la plus élevée dans la chaîne. Les fréquences des caractères sont évaluées en omettant les caractères segmentées, c'est-à-dire ceux pour lesquels $b(i-1) = b(i) = 1$.

Cette façon de faire nous assure de choisir a dans une région où se trouvent potentiellement beaucoup de sous-chaînes similaires. De plus, ceci fait en sorte que beaucoup de groupes de sous-chaînes seront générés lors de l'étape de segmentation-agrégation permettant ainsi à l'algorithme de se commettre sur un meilleur choix.

Naturellement, cette heuristique pour le choix de a ne garantit pas une solution optimale pour l'algorithme. Dans des situations où on souhaite se rapprocher de l'optimum, il est possible de randomiser cette étape en choisissant plusieurs a différents et en conservant celui produisant le meilleur groupe de sous-chaînes. Nous n'avons pas implanté cette variante et avons néanmoins obtenu de façon consistante de très bons résultats.

5.5.3 Boucle de segmentation-agrégation

Une fois la position initiale a choisie, l'algorithme procède à une boucle de segmentation-agrégation au cours de laquelle plusieurs groupes de sous-chaînes seront générés et évalués. Un seul de ces groupes sera retenu.

Segmentation

L'étape de segmentation consiste à sélectionner une sous-chaîne d'origine. Pour être valide, une telle sous-chaîne doit contenir la position a, être non segmentée, et ne pas dépasser une longueur maximale spécifiée par l'usager. Plus formellement, pour que $c[i, j]$ soit une sous-chaîne d'origine valide il faut que

$$\begin{aligned} &i \leq a \leq j \ \text{ et} \\ &b_{i-1} = b_i = ... = b_j = 0 \ \text{ et} \\ &j - i < \lambda \ , \end{aligned}$$

où λ est la longueur maximale de sous-chaîne spécifiée par l'usager.

Agrégation

L'étape d'agrégation consiste simplement à utiliser la sous-chaîne d'origine et la relation $\mathcal{M}$ décrite à la section 5.5.1 pour générer un groupe de sous-chaînes C. On utilise ensuite la largeur et la hauteur du groupe pour calculer son importance, nommée volume du groupe, de la façon suivante,

$$\mathrm{vol}(C) = (|C| - 1)\tilde{C}^{\rho}$$

où $\rho > 0$ est un paramètre spécifié par l'usager. Une petite valeur de ρ favorise les groupes de sous-chaînes plus hauts que larges, c'est-à-dire des groupes contenant un grand nombre de courtes sous-chaînes, et inversement pour une grande valeur de ρ. L'effet de ce paramètre est clairement visible à la figure 5.5. Un groupe ne contenant qu'une seule sous-chaîne est dit dégénéré et son volume est 0.

Itération

Les deux étapes précédentes sont répétées jusqu'à ce que toutes les sous-chaînes d'origine valides aient été sélectionnées. Il est facile de déterminer qu'au maximum, le nombre de sous-chaînes d'origine valides est de $\lambda(\lambda+1)/2-1$. Ainsi le nombre d'itérations de la boucle de segmentation-agrégation croît avec le carré de la longueur maximale spécifiée par l'usager. En pratique, cependant, cet ensemble est souvent beaucoup plus petit étant donné que le nombre de 1 dans le signal binaire b augmente rapidement au fur et à mesure de l'exécution de l'algorithme. Pour les exemples présentés ici nous utilisons toujours une valeur de $\lambda = 15$.

5.5.4 Sélection d'un groupe de sous-chaînes

Le résultat de la boucle précédente est un grand nombre de groupes de sous-chaînes potentiels. L'algorithme sélectionne, parmi ceux-ci, le groupe de volume maximal. Si tous les groupes sont dégénérés, alors l'algorithme sélectionne un groupe contenant une sous-chaîne de deux caractères. Ceci dans le but de ne pas segmenter inutilement des intervalles qui pourraient être utilisés de façon plus efficace dans les groupes de sous-chaînes qui seront créés lors des itérations suivantes.

Toutes les sous-chaînes du groupe sélectionné sont marquées comme étant segmentées en plaçant des 1 aux endroits appropriés de b. Il est possible ici d'apporter une optimisation à l'algorithme en retirant de la chaîne c toutes les sous-chaînes segmentées, c'est-à-dire les caractères correspondant à des séquences continues de 1 dans le signal b. Dans notre implantation, nous avons opéré cette contraction de c à chaque fois que le signal de partitionnement contenait plus de 25% de 1. Nous avons remarqué une augmentation importante des performances lorsque cette optimisation était utilisée.

La boucle principale est ensuite répétée et ce, jusqu'à ce que toute la chaîne ait été placée dans des groupes de sous-chaînes, c'est-à-dire que b ne contienne que des 1. À ce moment, il est possible de procéder aux deux opérations suivantes, qui visent respectivement à diminuer le nombre de groupes extraits et à limiter les différences de vitesses trop importantes entre les mouvements contenus dans un groupe.

Fusion des groupes dégénérés

Lorsque certains mouvements ne sont présents qu'une seule fois dans la banque de données, comme c'est souvent le cas, l'algorithme a tendance à produire un grand nombre de groupes

de sous-chaînes dégénérés composés de seulement deux caractères. En vue d'obtenir une représentation plus compacte, il est possible de fusionner tous les groupes dégénérés adjacents. Plus formellement, deux groupes dégénérés contenant respectivement les sous-chaînes $c[i, j]$ et $c[j, o]$ sont fusionnés en un seul groupe contenant la sous-chaîne $c[i, o]$.

Séparation des mouvements de vitesse variable

Comme nous l'avons mentionné à la section 5.4.4, on souhaite considérer deux mouvements comme similaires même s'ils s'exécutent à des vitesses différentes. Ceci peut avoir comme effet d'intégrer dans un même groupe de sous-chaînes des mouvements très rapides et très lents. Dans certaines applications, comme pour la compression, cet effet est désirable. Dans d'autres cas, comme le séquençage de mouvements présenté au chapitre 7, ces différences de vitesse peuvent causer des problèmes.

Dans ce cas, il est possible de séparer le groupe en fonction de la vitesse relative de chaque sous-chaîne. On peut estimer la vitesse relative de la sous-chaîne $c[i, j]$ simplement en divisant le nombre de poses qu'elle contient par la largeur du groupe de sous-chaînes, soit $\big(\bar{F}(j) - \bar{F}(i) + 1\big)/\tilde{C}$. Ensuite, si la variance des vitesses est trop importante, on procède à la séparation du groupe. Pour ce faire, nous avons utilisé la technique des k-moyennes (*k-means*) en déterminant le nombre de moyennes nécessaire à ce que chaque groupe ait une variance de vitesse inférieure au paramètre γ spécifié par l'usager. En pratique, nous n'avons jamais eu besoin de séparer un groupe de sous-chaînes en plus de deux sous-groupes. Pour tous les résultats présentés plus bas nous avons utilisé $\gamma = 1.5$.

5.6 Raffinement de la segmentation

La sortie de la boucle principale est un ensemble de groupes de sous-chaînes qui doivent maintenant être convertis en groupes de mouvements. Pour ce faire, il suffit de faire correspondre un indice de pose à chaque indice apparaissant au début ou à la fin d'une sous-chaîne. Ainsi, la sous-chaîne $c[i', j']$ sera convertie dans la sous-séquence $\mathbf{d}[i, j]$.

Nous utilisons une approche itérative qui améliore successivement cette correspondance. L'indice i' de la chaîne est tout d'abord associé à i_0, puis l'assignation est raffinée pour donner i_1, i_2, et ainsi de suite jusqu'à l'obtention d'une assignation satisfaisante. Pour débuter, nous utilisons $i_0 = \bar{F}(i')$. Autrement dit, chaque caractère est d'abord associé au centre des poses qu'il représente. Ceci nous permet de convertir, en première approximation, tous les groupes

de sous-chaînes C en autant de groupes de mouvements M. La correspondance est cependant assez grossière et rien ne garantit que les sous-séquences d'un groupe soient bien alignées au début ou à la fin.

Pour trouver i_1 remarquons tout d'abord que, si on s'intéresse à tous les groupes de mouvements, alors un indice i_0 se trouve dans exactement deux sous-séquences : $\mathbf{d}[i_0^-, i_0]$ et $\mathbf{d}[i_0, i_0^+]$.[4] Ces sous-séquences se trouvent à leur tour respectivement dans les groupes M^- et M^+. On souhaite déterminer un indice i_1 correspondant à une pose qui s'aligne le mieux possible avec toutes les poses finales des sous-séquences de M^- et toutes les poses initiales des sous-séquences de M^+.

Pour ce faire, nous évaluons $\bar{\mathbf{d}}^-$ la moyenne des poses finales de M^- et $\bar{\mathbf{d}}^+$ celle des poses initiales de M^+. Notre but est ainsi d'obtenir i_1 dans le voisinage de i_0 de façon à le rapprocher le plus possible de ces deux moyennes. On doit aussi s'assurer que $i_0^- < i_1 < i_0^+$ de façon à conserver des sous-séquences valides.

Plus généralement, à l'itération $o+1$ on modifie i la façon suivante,

$$i_{o+1} = \underset{i_o^- < i_{o+1} < i_o^+}{\operatorname{argmin}} \; |\mathbf{d}(i_{o+1}) - \bar{\mathbf{d}}^-| + |\mathbf{d}(i_{o+1}) - \bar{\mathbf{d}}^+| + \beta |i_{o+1} - i_0| \, \frac{1}{n} \sum_{j=1}^{n} |\mathbf{d}(j) - \bar{\mathbf{d}}| \; ,$$

où $\frac{1}{n} \sum_{j=1}^{n} |\mathbf{d}(j) - \bar{\mathbf{d}}|$ est la distance moyenne entre une pose de la banque de données et la pose moyenne. Le terme $\beta |i_{o+1} - i_0|$ vise à s'assurer qu'on ne s'éloigne pas trop de l'assignation initiale, évitant ainsi une dérive de l'algorithme. Le paramètre β, spécifié par l'usager, permet de contrôler ce comportement. En pratique, on utilise toujours $\beta = 0.1$. Nous résolvons ce problème de minimisation discrète par une simple recherche linéaire.

Chaque itération est effectuée sur tous les indices apparaissant dans un groupe de mouvements. L'ordre dans lequel les indices sont traités peut avoir une influence sur le résultat de cet algorithme. Nous les parcourons donc dans un ordre aléatoire et différent à chaque itération.

Nous n'avons pas étudié les propriétés générales de convergence de cet algorithme. Cependant, pour tous les résultats présentés plus bas, l'algorithme converge vers un ensemble d'indices stable. En général, cette convergence est atteinte en moins de 20 itérations.

Suite au raffinement de la segmentation on obtient un ensemble de groupes de mouvements,

$$\{M(1), M(2), ...\} \; .$$

[4] Sauf pour les sous-séquences se trouvant au début ou à la fin d'une animation, pour lesquels i_0 n'apparaît que dans une seule sous-séquence. De telles assignations ne sont jamais modifiées (*i.e.* $i_o = i_0$) de façon à s'assurer que les sous-séquences sont alignées sur le début ou la fin des animations.

α	Pourcentage d'énergie conservée lors de la *PCA* (section 5.4.2)
κ	Nombre de grappes pour l'agrégation, taille de l'alphabet (section 5.4.3)
τ	Distance maximale entre deux grappe similaires, déterminé par τ' (section 5.4.6)
τ'	Quantile utilisé pour déterminer τ (section 5.4.6)
ρ	Importance de la largeur d'un groupe dans le calcul du volume (section 5.5.3)
λ	Longueur maximale des sous-chaînes considérées (section 5.5.3)
γ	Variance maximale des vitesses des mouvements dans un groupe (section 5.5.4)
β	Tension pour l'ajustement précis des poses (section 5.6)

TAB. 5.1 – Paramètres spécifiés par l'usager pour l'algorithme de détection de sous-séquences redondantes.

Comme pour les groupes de sous-chaînes, on définit la hauteur du groupe de mouvements $M(i)$ comme étant le nombre de sous-séquences qu'il contient soit $|M(i)|$. La largeur du groupe est définie comme le nombre moyen d'échantillons dans toutes les sous-séquences du groupe,

$$\tilde{M} = \frac{1}{|M|} \sum_{j=1}^{|M|} n_j^{\mathtt{s}} \ ,$$

où $n_j^{\mathtt{s}}$ dénote le nombre d'échantillons dans la j-ième sous-séquence. On définit aussi le volume d'un groupe de mouvements,

$$\mathrm{vol}(M) = (|M| - 1)\tilde{M}^{\rho} \ . \tag{5.2}$$

Ici, $\rho > 0$ est le paramètre défini à la section 5.5.3.

5.7 Paramètres de contrôle

La description de l'algorithme qui précède introduit un certain nombre de paramètres de contrôle qui doivent être spécifiés par l'utilisateur. Ces différents paramètres sont repris et brièvement décrits au tableau 5.1.

À première vue, on pourrait craindre que le grand nombre de paramètres associés à la technique n'en rende l'ajustement complexe et délicat. Cependant, beaucoup de ces paramètres n'ont pas à être modifiés en fonction des caractéristiques de de la banque de données d'animations traitée. Ainsi, pour tous les résultats présentés plus bas, nous n'avons pas eu à modifier les valeurs de α, τ', λ, γ et β. Quant au paramètre τ, qui varie en fonction des données, il est déterminé automatiquement à partir de τ'. Les valeurs utilisées pour ces différents paramètres sont spécifiées dans les sections correspondantes.

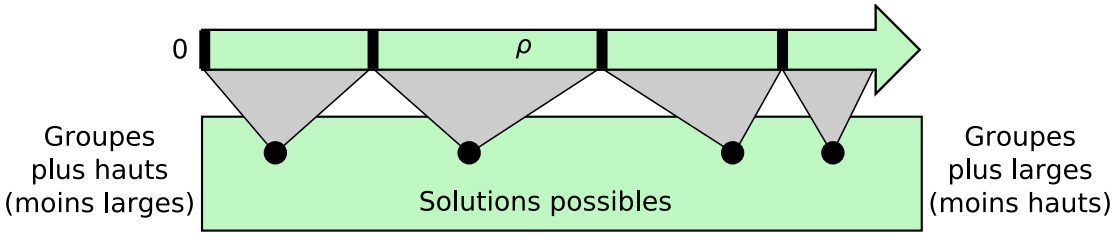


FIG. 5.13 – Cette figure montre comment une modification au paramètre ρ permet de basculer d'une solution à une autre. Lorsque ρ est faible, les solutions obtenues contiennent des groupes de mouvements plus hauts et moins larges, et inversement quand ρ est élevé. Une modification à ρ ne permettra cependant jamais l'apparition d'une solution contenant des groupes à la fois moins hauts et moins larges.

Les deux seuls paramètres qui doivent être ajustés en fonction des données et des désirs de l'utilisateur sont κ et ρ. Le premier est un paramètre que l'on retrouve dans beaucoup d'applications faisant appel à l'agrégation. En général, une application utilisant l'agrégation dans le but de classifier des données sera très sensible à la valeur κ choisie. Dans notre cas, cependant, le but de l'agrégation est de permettre une représentation simplifiée de la banque de données d'animations et non d'identifier de réelles grappes présentes dans les données. Une variation de κ se traduira ainsi par une tolérance plus ou moins grande pour la distance entre des poses jugées semblables. En pratique nous avons remarqué qu'il était possible d'obtenir des résultats de très bonne qualité même en faisant substantiellement varier κ, et ce pour toutes les banques de données considérées.

Le paramètre ρ, quant à lui, a été ajouté pour accroître le contrôle de l'utilisateur. Dans un certain sens, ce paramètre influence l'allure de la solution retenue mais pas sa qualité. Ceci vient du fait que, peu importe la valeur de ρ, l'augmentation de la hauteur ou de la largeur d'un groupe de mouvements se traduira par l'augmentation de son volume (voir équation 5.2). En fait, une modification de ρ peut permettre de basculer d'une solution à une autre, mais ne permet pas d'atteindre des solutions intermédiaires de moins bonne qualité. Ceci est représenté graphiquement à la figure 5.13.

5.8 Résultats

Nous avons testé l'algorithme décrit précédemment sur un grand nombre de banques de données différentes, la plus importante comportant 7300 échantillons acquis à une fréquence de 120 Hz pour un total d'environ 1 minute d'animation. L'ensemble complet des données utilisées totalise environ 10 minutes d'animation.

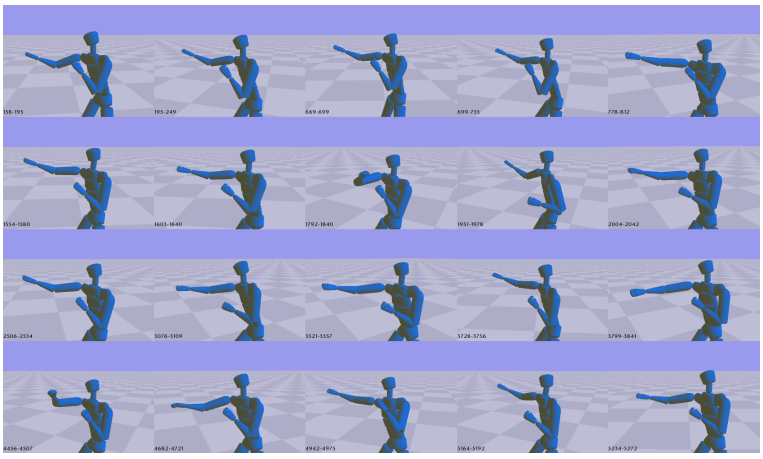
Certaines des banques de données de test ne contiennent qu'une seule activité : un acteur qui boxe, qui effectue différents exercices ou qui se déplace en marchant le long d'une trajectoire arbitraire. D'autres collections contiennent des animations représentant des mouvements distincts et parfois non reliés. Toutes les animations sont extraites de la *CMU Motion Capture Database* [CMU] et utilisent une fréquence d'échantillonnage de 120 Hz. Quelques mouvements contiennent des poses légèrement erronnées, typiques des données brutes acquises par capture de mouvements. Tous les tests ont été effectués sur un processeur Athlon64 3500+ muni de 2 GB de mémoire vive. Notre algorithme a été implanté à l'aide du logiciel *Matlab*.

L'étape la plus coûteuse de l'extraction des groupes de mouvements est l'agrégation des poses. Ce processus s'exécute en 460 secondes pour la banque de données de 7300 échantillons lorsque $\kappa = 250$ grappes. La suite de l'algorithme de partitionnement en groupes de mouvements s'exécute en 76 secondes sur cet exemple. Pour une animation de marche de 1900 échantillons, le partitionnement s'exécute en 4 secondes.

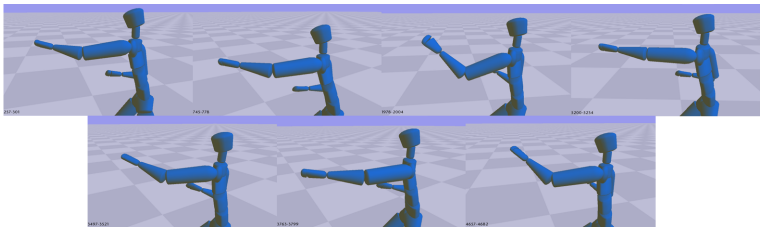
La structure finale obtenue demande très peu d'espace mémoire, requérant uniquement le stockage des indices des poses de début et de fin de chaque sous-séquence. La taille totale des groupes de mouvements pour toutes les banques de données s'élève à environ 100 KB d'espace disque, et ce, dans un format texte non compressé.

Il est difficile d'évaluer quantitativement la qualité des résultats obtenus étant donné que nous ne disposons pas d'une solution de référence avec laquelle nous pourrions effectuer des comparaisons. Nous nous inspirons donc de certains travaux en extraction de mouvements [KG04b] et procédons à une évaluation visuelle des groupes de mouvements ainsi que de la banque de données. Ceci nous permet de comparer les groupes extraits par l'algorithme de ceux que nous aurions identifiés manuellement.

Le contenu d'un groupe de mouvements est difficile à illustrer sous forme imprimée, nous encourageons d'ailleurs le lecteur à visionner le vidéo associé à cette thèse [Bea07b]. La figure 5.14 tente néanmoins d'illustrer deux groupes de mouvements extraits d'une banque de données contenant des animations de boxe. La banque de données initiale, correspondant au



(a) Coups de poing droit.



(b) Coups de poing gauche.

FIG. 5.14 – Deux des groupes de mouvements extraits par l'algorithme sur une banque de données d'animations de boxe. Chaque image correspond à une sous-séquence dont seule la pose centrale est illustrée. (a) Un groupe de mouvements comprenant des coups de poing droit. (b) Un groupe comprenant des coups de poing gauche.

fichier 14_02.amc dans la *CMU Motion Capture Database* [CMU], contient 5423 échantillons comprenant un assortiment de coups de poing, de mouvements d'évitement et de jeux de pieds. Les résultats présentés ont été obtenus en utilisant $\rho = 1$.

Le premier groupe de mouvements contient 20 coups de poing droit de 36 échantillons en moyenne. Le second groupe contient 7 coups de poing gauche de 32 échantillons en moyenne. Parallèlement, l'algorithme a aussi extrait un certain nombre de groupes de mouvements non dégénérés regroupant différents jeux de pieds ou des mouvements d'attente neutres. Nous nous attardons cependant sur les coups de poing compte tenu qu'ils sont facilement repérables manuellement.

Une inspection visuelle de la banque de données permet d'identifier 22 coups de poing droit et 11 coups de poing gauche. Le style de ces coups varie bien que la plupart représentent de rapides *jabs* de différentes forces. Les versions plus stylisées sont moins fréquentes et sont généralement plus longues. Par exemple, on peut observer deux coups droits de style "marteau"[5] environ deux fois plus longs que les *jabs*.

La valeur de ρ choisie affecte les groupes de mouvements extraits. Par exemple, $\rho = 2$ favorise la segmentation de mouvements plus longs. Dans la séquence de boxe, ceci se traduit par l'apparition de groupes de mouvements contenant des combinaisons et des coups stylisés plus longs. Un groupe contient 6 combinaisons gauche-droite (69 échantillons), un autre 3 *uppercuts* droits (89 échantillons), 2 coups droits de style "marteau" (51 échantillons), 4 coups droits puissants (42 échantillons), 5 coups droits faibles (23 échantillons). Ce dernier groupe contient aussi un coup gauche constituant un résultat erronné de l'algorithme. Pour chaque groupe nous indiquons entre parenthèses le nombre moyen d'échantillons. Les différentes autres valeurs de ρ utilisées donnaient systématiquement une de ces deux solutions, si on fait abstraction des légères différences introduites par les aspects aléatoires de l'algorithme.

En vue de tester la sensibilité de l'algorithme en présence de bruit, nous l'avons exécuté avec $\rho = 1$ sur une version bruitée de l'animation de boxe. Pour ce faire, nous avons perturbé chaque échantillon de chaque degré de liberté par un bruit gaussien indépendant de moyenne nulle et de variance 0.1. La structure des groupes de mouvements ainsi extraite est très similaire à celle obtenue sans l'ajout de bruit bien que quelques différences mineures apparaissent. Par exemple, le groupe des coups de poing droit contient maintenant 19 mouvements (41 échantillons) et celui des coups de poing gauche en contient 7 (30 échantillons).

[5] Nous avons baptisé "marteau" un coup où le poing s'abat brusquement de haut en bas.

Nous avons aussi utilisé l'algorithme sur différentes autres banques de données et avons obtenu des résultats similaires. Certains de ces résultats sont présentés au chapitre 7 sous la forme de graphes de groupes de mouvements. En particulier, nous avons combiné deux animations différentes en une unique banque de données et l'algorithme est parvenu à extraire les mêmes groupes de mouvements que sur les animations traitées indépendamment. Le résultat comprenait néanmoins de légères différences dues aux aspects aléatoires de l'algorithme.

5.9 Conclusion

Notre principale contribution, dans ce chapitre, est une technique de segmentation automatique basée sur l'identification de sous-séquences redondantes à l'intérieur d'une banque de données. Cette technique s'appuie sur une approche originale combinant la segmentation et l'agrégation de mouvements à l'intérieur d'un processus itératif. Ceci est rendu possible par une représentation simplifiée de la banque de données à l'aide d'une chaîne de caractères ainsi que par une technique robuste et efficace de comparaison de sous-chaînes basée sur l'utilisation d'une matrice de similarité.

Une des limitations techniques de notre approche est liée à l'utilisation d'un algorithme d'*Expectation Maximization* requérant la présence en mémoire de toutes les données à agréger. Ceci a causé des problèmes de débordement de mémoire dans les banques de données plus importantes que nous avons tenté de traiter. Nous envisageons différentes solutions à ce problème, en particulier le sous-échantillonnage des données ou l'utilisation d'une technique d'agrégation en-ligne (*online clustering*).

Notons aussi, bien que nous nous assurons que les mouvements présents dans un même groupe soient alignés au début et à la fin, rien ne garantit que leurs poses intermédiaires soient bien alignées. À cette fin, nous pourrions utiliser une courbe d'enregistrement [KG03], mais en pratique ce genre de précision dans l'alignement ne s'est pas avéré nécessaire.

Les groupes de mouvements obtenus à l'aide de l'algorithme décrit dans ce chapitre se rapprochent de plusieurs structures utilisées dans différentes applications. En particulier, ils constituent une représentation des cohérences temporelles à grande échelle que nous exploitons, au chapitre 6, dans le contexte de la compression d'animations. Au chapitre 7, nous montrons comment ces groupes de mouvements peuvent mener à des structures similaires aux graphes de mouvements paramétriques [HG07] et ainsi permettre de synthétiser de nouvelles animations.

Chapitre 6

Compression de sous-séquences redondantes

Well, I do not want any of those things. What I chiefly want is information.
Jonathan Strange

Le chapitre précédent introduit une technique permettant d'identifier automatiquement des groupes de sous-séquences similaires à l'intérieur d'une banque de données acquises par capture de mouvements. Si on peut intuitivement penser que de telles structures ouvrent la porte au développement de techniques de compression plus efficaces, il n'en demeure pas moins nécessaire d'étudier les différentes façons de les exploiter.[1]

Comme nous l'avons déjà mentionné, la séparation d'une banque de données en groupes de mouvements peut être vue comme la création d'un dictionnaire, chaque groupe représentant une entrée du dictionnaire fréquemment répétée dans la banque de données. Ce chapitre s'attache à cette interprétation pour mettre au point une technique de compression inspirée des méthodes par dictionnaire dynamique.

Après une brève introduction aux techniques classiques de compression par dictionnaire dynamique, nous montrons comment les groupes de mouvements peuvent être utilisés pour mettre au point une compression par dictionnaire avec pertes. Pour ce faire, nous réutilisons et étendons certaines des techniques présentées au chapitre 4. Nous décrivons ensuite en détail le format de compression utilisé et nous analysons les résultats obtenus. Finalement, nous discutons des extensions potentielles en s'intéressant particulièrement à l'exploitation possible des corrélations entre les degrés de liberté.

[1] Dans ce chapitre, r, n, k et $\mathbf{d}$ désignent respectivement $r^{\mathtt{W}}$, $n^{\mathtt{M}}$, $k^{\mathtt{M}}$ et $\mathbf{d}^{\mathtt{M}}$.

6.1 Introduction

L'expression compression par dictionnaire (*dictionary coder* ou *substitution coder*) est généralement réservé aux techniques de compression sans pertes qui, dans un premier temps, décomposent la chaîne de symboles à encoder en une série de sous-chaînes. Chacune de ces sous-chaînes est ensuite représentée par un simple indice référant à une entrée dans la structure de dictionnaire connue aussi bien de l'encodeur que du décodeur.

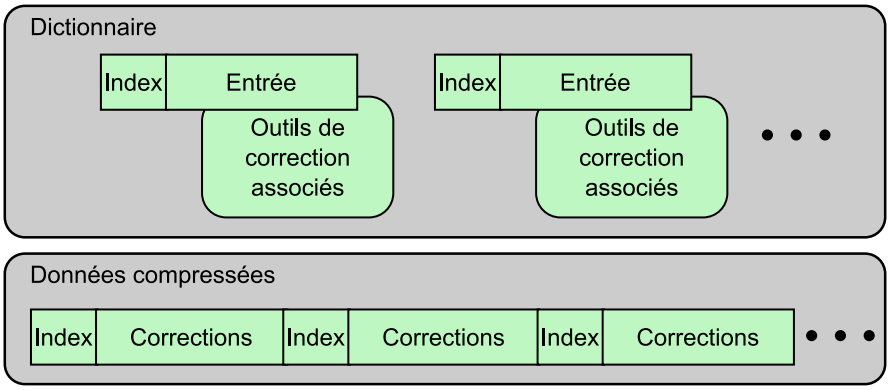
Les techniques utilisant un dictionnaire statique et constant pour l'encodage de n'importe quelle chaîne de symboles sont généralement jugées de peu d'intérêt. En effet, aucun dictionnaire ne peut être efficace dans toutes les situations : un texte en français n'a pratiquement rien en commun avec un texte en grec ancien ou avec le fichier binaire représentant un logiciel. En guise d'alternative, on pourrait proposer d'utiliser un dictionnaire statique mais adapté à une situation particulière. Ceci oblige cependant l'encodeur et le décodeur à partager un dictionnaire approprié, ou alors à se le transmettre avant de pouvoir s'échanger la chaîne de symboles compressée.

Une compression par dictionnaire dynamique utilise plutôt un dictionnaire différent pour l'encodage de chaque chaîne de symboles. Ce dictionnaire est construit au fur et à mesure de la transmission et ne contient ainsi que des entrées qui seront utiles dans le cadre de la chaîne en cours de compression. Les premières techniques de compression par dictionnaire dynamique ont été introduites par Lempel et Ziv [ZL77, ZL78].

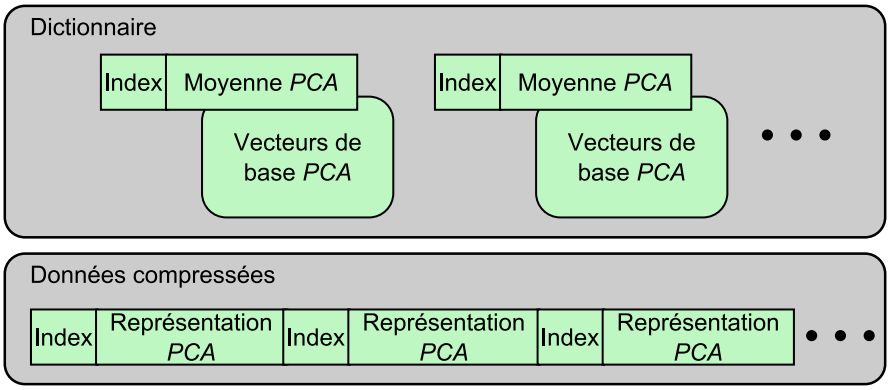
6.1.1 Encodage par dictionnaire avec pertes

Il est possible d'étendre un peu la définition d'une méthode par dictionnaire pour l'utiliser dans le contexte d'une compression avec pertes. Dans un tel cas, et pour procéder par analogie, une sous-séquence de symboles peut être représentée par une entrée du dictionnaire même si celle-ci comporte quelques erreurs. Par la suite, il peut être souhaitable d'inclure de l'information permettant de corriger partiellement ces erreurs. Pour faciliter cette correction, l'entrée du dictionnaire peut être munie d'un certain nombre d'informations supplémentaires. Cette définition est illustrée graphiquement à la figure 6.1(a).

La compression de banques de données proposée par Arikan [Ari06], détaillée à la section 3.4.4, peut ainsi être interprétée comme une méthode de compression par dictionnaire avec pertes. Rappelons que, dans cette technique, la banque de données est tout d'abord divisée en segments de tailles constantes. Chaque segment est approximé à l'aide d'une trajectoire cubique



(a) Compression générale par dictionnaire avec pertes



(b) La technique de Arikan [Ari06] vue comme une compression par dictionnaire avec pertes.

FIG. 6.1 – Compression par dictionnaire avec pertes.

qui peut être encodée dans un vecteur de très haute dimensionnalité. Ces vecteurs sont ensuite agrégés et une base de plus faible dimensionnalité, obtenue par *PCA*, est attachée à chacune des grappes ainsi formées. Finalement, chaque segment est encodé par l'indice de sa grappe suivi d'une représentation du vecteur de la trajectoire dans la base *PCA*.

Pour interpréter cette technique comme une méthode par dictionnaire, on considère chaque grappe comme une entrée du dictionnaire. La moyenne des vecteurs de la grappe, calculée lors de la *PCA*, constitue alors une approximation de la trajectoire associée à cette entrée. Les vecteurs de la base *PCA* permettent de corriger, dans une certaine mesure, les erreurs introduites par cette approximation. Cette interprétation est illustrée à la figure 6.1(b).

Un des principaux problèmes de cette technique, suivant cette interprétation, provient du fait que les segments sont de taille fixe, typiquement de 16 à 32 échantillons par segment. Cependant, rien ne garantit que les sous-séquences redondantes présentes dans la banque de données seront alignées suivant de telles frontières. Il est donc possible que deux gestes numériquement très similaires correspondent, suite à la segmentation imposée par Arikan, à des vecteurs de trajectoire forts différents et donc à des entrées distinctes du dictionnaire.

À l'opposé, l'extraction automatique de groupes de mouvements proposée au chapitre 5 permet d'identifier des sous-séquences redondantes bien alignées. Une des contributions présentées dans ce chapitre consiste à montrer que de telles structures permettent de créer un dictionnaire compact. Nous proposons aussi une méthode, inspirée de la compression par ondelettes du chapitre 4, permettant de corriger efficacement les erreurs d'approximation introduites par l'utilisation du dictionnaire.

6.2 Compression des groupes de mouvements

La technique de compression que nous proposons est inspirée du modèle de compression par dictionnaire avec pertes, illustré à la figure 6.1(a), où les entrées du dictionnaire sont les groupes de mouvements. Cependant, à la différence de ce modèle, nous ne spécifions pas explicitement un ensemble d'outils de correction pour chaque entrée. En effet, nous proposons une nouvelle approche, que nous avons nommée *PCA quantifiée*, qui permet simultanément de spécifier les corrections et de construire l'ensemble des outils de correction.

Nous justifions cette nouvelle approche en soulignant dans un premier temps les problèmes liés à l'utilisation de la *PCA* standard. Par la suite, nous expliquons en détail les fondements théoriques et l'algorithme permettant de réaliser la *PCA* quantifiée.

6.2.1 Problèmes de l'approche par *PCA*

Un premier réflexe pourrait être de simplement remplacer l'étape de segmentation uniforme dans la compression d'Arikan [Ari06] par notre technique d'extraction de groupes de mouvements. Autrement dit, on pourrait construire une base *PCA* pour chaque groupe de mouvements et exprimer les données dans ces bases.

Une première différence, cependant, vient du fait que les dictionnaires générés par la méthode d'Arikan ne contiennent que de 1 à 20 entrées pour des banques de données de plus d'une heure. En contrepartie, les dictionnaires obtenus par groupes de mouvements peuvent compter quelques dizaines d'entrées pour des animations de quelques minutes, et ce nombre d'entrées risque d'augmenter de façon importante pour des banques de données d'une heure ou plus. Ainsi, le nombre de mouvements correspondant à une entrée est beaucoup plus important dans la compression proposée par Arikan.

Une autre différence vient du fait que les segments d'Arikan sont très courts, de 16 à 32 échantillons, et que chaque trajectoire s'exprime simplement par les quatre points de contrôle d'une cubique. Dans notre cas, les segments peuvent être beaucoup plus longs et requièrent généralement beaucoup de coefficients pour être exprimés correctement.

Ces deux différences ont une incidence directe sur l'efficacité d'une approche par *PCA* directe. En effet, dans le cas d'Arikan, une base *PCA* correspondant à une entrée du dictionnaire est réutilisée pour un très grand nombre de mouvements. De plus, comme les segments sont courts, la dimensionnalité des vecteurs de base reste relativement faible. Ainsi, le coût de stockage de la base est amorti et devient pratiquement négligeable. Dans notre cas, ce coût reste très important, au point où il est presque toujours plus avantageux d'encoder directement chacun des mouvements plutôt que de tenter de stocker une base *PCA*.

6.2.2 Projection *PCA* quantifiée

Nous proposons maintenant une technique de projection, que nous nommons *PCA* quantifiée, et qui permet de contourner le coût supplémentaire (*overhead*) requis pour le stockage d'une base *PCA* standard. L'idée consiste à relaxer légèrement le critère d'optimalité de la *PCA* tout en s'assurant que le stockage de la base permette simultanément d'encoder des données.

Comme nous l'avons mentionné à la section 3.1.2, la *PCA* standard est optimale au sens des moindres carrés. C'est-à-dire que si $\boldsymbol{\mu}$ est la moyenne des données et que $\mathbf{e}(1)$ est le premier

vecteur de la base *PCA*, alors la projection des données sur ce vecteur est maximale :

$$\sum_i \langle \mathbf{d}(i) - \boldsymbol{\mu}, \mathbf{e}(1) \rangle^2 \ \geq \ \sum_i \langle \mathbf{d}(i) - \boldsymbol{\mu}, \mathbf{x} \rangle^2, \ \forall \mathbf{x} \ \text{ t.q. } \ |\mathbf{x}| = 1, \tag{6.1}$$

où les $\mathbf{d}(i)$ sont les données à projeter. En général, pour le o-ième vecteur de base, l'optimalité de la *PCA* garantit que

$$\sum_i \left\langle \text{proj} \left(E_{o-1}^{\perp}, \mathbf{d}(i) - \boldsymbol{\mu} \right), \mathbf{e}(o) \right\rangle^2 \ \geq \ \sum_i \left\langle \text{proj} \left(E_{o-1}^{\perp}, \mathbf{d}(i) - \boldsymbol{\mu} \right), \mathbf{x} \right\rangle^2 , \tag{6.2}$$

et ce pour tout vecteur $\mathbf{x}$ unitaire. Dans cette équation, le sous-espace linéaire E_{o-1} est défini par $\text{span}\{\mathbf{e}(1), \mathbf{e}(2), ..., \mathbf{e}(o-1)\}$. La projection dans le complément orthogonal de cet espace, $E_{o-1}^{\perp}$, peut être réécrit,

$$\text{proj} \left(E_{o-1}^{\perp}, \mathbf{d}(i) - \boldsymbol{\mu} \right) = \mathbf{d}(i) - \boldsymbol{\mu} - \text{proj} \left(E_{o-1}, \mathbf{d}(i) - \boldsymbol{\mu} \right) \ . \tag{6.3}$$

Dans le cas où les vecteurs sur lesquels on projette constituent une base orthonormale, comme c'est le cas avec la *PCA* standard, alors on peut définir la projection par une somme de produits scalaires,

$$\text{proj}(E_{o-1}, \mathbf{d}(i) - \boldsymbol{\mu}) = \sum_{j=1}^{o-1} \left\langle \mathbf{d}(i) - \boldsymbol{\mu}, \mathbf{e}(j) \right\rangle \mathbf{e}(j) \ . \tag{6.4}$$

Lorsqu'on utilise la *PCA* pour la compression, le coût supplémentaire vient du fait que le stockage des vecteurs $\mathbf{e}(j)$ ne contribue pas directement à l'encodage des données. Nous proposons donc plutôt de forcer les vecteurs de base à être choisis parmi les données, et ce au risque d'obtenir une base légèrement moins optimale. Cette limitation de la *PCA* à un ensemble fini de vecteurs explique l'appellation *PCA* quantifiée. Ainsi, $\boldsymbol{\mu}$ devra être le vecteur contenu dans l'ensemble des données $D = \{\mathbf{d}(1), ..., \mathbf{d}(n)\}$ qui correspond le plus précisément à la moyenne,

$$\boldsymbol{\mu} = \underset{\mathbf{x} \in D}{\text{argmin}} \left| \mathbf{x} - \frac{1}{n} \sum_{i=1}^{n} \mathbf{d}(i) \right|^2 \ . \tag{6.5}$$

Le vecteur $\mathbf{e}(1)$ est choisi de façon similaire, en utilisant le critère d'optimalité de l'équation 6.1 limité à l'ensemble des vecteurs de base,

$$\mathbf{e}(1) = \underset{\mathbf{x} \in D'}{\text{argmax}} \ \sum_i \left\langle \mathbf{d}(i) - \boldsymbol{\mu}, \frac{\mathbf{x}}{|\mathbf{x}|} \right\rangle^2 ,$$

où D' est l'ensemble de vecteurs obtenus en soustrayant la moyenne à chaque donnée,

$$D' = \{ \mathbf{d}(i) - \boldsymbol{\mu} | \ i \leq n \} \ .$$

De la même façon, on peut trouver le o-ième vecteur de base en adaptant le critère de l'équation 6.2 au cas de la *PCA* quantifiée,

$$\mathbf{e}(o) = \underset{\mathbf{x} \in D'}{\operatorname{argmax}} \sum_i \left\langle \operatorname{proj}(E_{o-1}^{\perp}, \mathbf{d}(i) - \boldsymbol{\mu}), \frac{\mathbf{x}}{|\mathbf{x}|} \right\rangle^2 . \tag{6.6}$$

La projection dans $E_{o-1}^{\perp}$ peut ici encore se faire à l'aide de l'équation 6.3. On ne peut cependant pas utiliser l'équation 6.4 étant donné que les vecteurs $\{\mathbf{e}(1), ..., \mathbf{e}(o-1)\}$ ne sont pas nécessairement orthogonaux. On peut cependant reformuler le problème de la projection dans E_{o-1} sous la forme d'un problème de moindres carrés.

Comme cette projection se trouve dans $\operatorname{span}\{\mathbf{e}(1), ..., \mathbf{e}(o-1)\}$, on sait qu'il existe $\mathbf{v} \in \mathbb{R}^{o-1}$ tel que

$$\operatorname{proj}(E_{o-1}, \mathbf{d}(i) - \boldsymbol{\mu}) = \mathbf{E}_{o-1}\, \mathbf{v} \ , \tag{6.7}$$

où $\mathbf{E}_{o-1} = \left[\ \mathbf{e}(1)\ \mathbf{e}(2)\ \cdots\ \mathbf{e}(o-1)\ \right]$, soit la matrice contenant un vecteur de base par colonne. De plus, comme la projection se fait perpendiculairement au sous-espace $\operatorname{span}\{\mathbf{e}(1), ..., \mathbf{e}(o-1)\}$, on sait que $\mathbf{v}$ minimise l'équation suivante,

$$\left| \ (\mathbf{d}(i) - \boldsymbol{\mu}) - \mathbf{E}_{o-1}\, \mathbf{v} \ \right|^2 .$$

La solution à ce genre de problème s'obtient en prenant la pseudo-inverse de la matrice,

$$\mathbf{v} = \mathbf{E}_{o-1}^{\mathsf{T}} (\mathbf{E}_{o-1} \mathbf{E}_{o-1}^{\mathsf{T}})^{-1} (\mathbf{d}(i) - \boldsymbol{\mu}) \ . \tag{6.8}$$

Le calcul de la base *PCA* quantifiée et le processus de projection est ainsi complètement défini par les équations 6.3 et 6.5 à 6.8.

6.2.3 Utilisation de la *PCA* quantifiée

Maintenant que nous avons défini une projection où la base est tirée des données, il reste à décrire une technique permettant de l'utiliser efficacement dans le cadre de la compression. En particulier, nous nous intéressons à la façon de compresser un seul groupe de mouvements, la compression de la banque de données entière est discutée plus loin. Pour mettre au point cette compression, nous proposons de réutiliser certains des résultats présentés au chapitre 4.

Soit le groupe de mouvements à compresser,

$$M = \left\{ \mathbf{d}^{\mathtt{I}}[i_1, j_1], \mathbf{d}^{\mathtt{I}}[i_2, j_2], ..., \mathbf{d}^{\mathtt{I}}[i_n, j_n] \right\} \ ,$$

pour lequel on note $n_o^{\mathtt{S}}$ le nombre d'échantillons dans la o-ième sous-séquence. En général ces sous-séquences sont toutes de longueurs différentes, donc les $n_o^{\mathtt{S}}$ sont tous différents. Pour

simplifier le traitement simultané de ces sous-séquences, nous procédons tout d'abord à un rééchantillonnage pour qu'elles comptent toutes exactement $n^{\mathtt{S}} = \max_o\{n_o^{\mathtt{S}}\}$ échantillons. Cette opération de sur-échantillonnage n'introduit pas de pertes significatives.

Il faut maintenant établir l'espace vectoriel dans lequel seront exprimées les données avant l'application de la *PCA* quantifiée. On souhaite construire un unique vecteur pour chaque mouvement. Une première approche pourrait être de construire, pour chaque sous-séquence, un vecteur de dimension $k^{\mathtt{I}}n^{\mathtt{S}}$ en mettant bout-à-bout les $n^{\mathtt{S}}$ vecteurs de pose qui comptent chacun $k^{\mathtt{I}}$ dimensions. Ceci conduit cependant à une dimensionnalité trop grande pour être utilisée efficacement lors de la *PCA* quantifiée.

Pour trouver un espace de plus faible dimensionnalité, nous proposons d'exploiter les importantes corrélations temporelles présentes dans les données. On applique ainsi à chaque sous-séquence la transformation en ondelettes décrite à la section 4.4.2. Ceci introduit un grand nombre de coefficients nuls dans la représentation des signaux. Si un coefficient donné est nul pour toutes les sous-séquences alors il est tout simplement omis ; ainsi les sous-séquences seront compactées de manière à s'exprimer à l'aide de $n^{\mathtt{C}}$ coefficients. Ces coefficients sont ensuite simplement concaténés pour créer un vecteur de dimension $k = k^{\mathtt{I}}n^{\mathtt{C}}$. En général, $n^{\mathtt{C}}$ est beaucoup plus petit que $n^{\mathtt{S}}$, ce qui réduit substantiellement la dimensionnalité des vecteurs. Au final, on obtient un vecteur $\mathbf{d}(i)$ de dimension k pour chacune des n sous-séquences.

La projection par *PCA* quantifiée peut ensuite être appliquée. Tout d'abord, on utilise l'équation 6.5 pour déterminer $\boldsymbol{\mu}$, ce qui correspond à identifier une sous-séquence $\mathbf{d}(i_{\boldsymbol{\mu}})$. Ceci nous permet de calculer l'ensemble D' en soustrayant $\boldsymbol{\mu}$ de tous les éléments $\mathbf{d}(i)$. Finalement, on applique l'équation 6.6 autant de fois que nécessaire. À chaque vecteur de base $\mathbf{e}(j)$ obtenu de la sorte correspond une sous-séquence $\mathbf{d}(i_j)$. En pratique, le nombre n de sous-séquences dans le groupe est beaucoup plus petit que le nombre de dimensions k. On obtiendra donc $n-1$ vecteurs de base $\mathbf{d}(i_j)$.[2] On résout les optimisations des équations 6.5 et 6.6 en effectuant simplement une recherche linéaire à travers l'ensemble des vecteurs admissibles.

Pour chacun des vecteurs de base $\mathbf{e}(j)$ on dispose aussi, via l'équation 6.8, d'un vecteur $\mathbf{v}(j)$ indiquant les composantes de $\mathbf{e}(j)$ le long des vecteurs de base précédents. Le nombre d'éléments dans le vecteur $\mathbf{v}(j)$ est $j-1$.

[2]On compte $n-1$ vecteurs de base car une des n sous-séquences a été utilisée pour produire $\mathbf{d}(i_{\boldsymbol{\mu}})$.

6.2.4 Compression par *PCA* quantifiée

La section précédente explique comment appliquer la *PCA* quantifiée sur un groupe de mouvements. Nous montrons maintenant comment l'utiliser pour développer une compression efficace. Cette compression sera réalisée en encodant tout d'abord $\mathbf{d}(i_{\boldsymbol{\mu}})$, puis successivement $\mathbf{d}(i_1)$, $\mathbf{v}(2)$, $\mathbf{d}(i_2)$, $\mathbf{v}(3)$, $\mathbf{d}(i_3)$, etc. L'avantage de cette compression vient du fait que chaque nouveau vecteur $\mathbf{d}(i_j)$ est approximé de plus en plus précisément par les vecteurs précédents.

L'encodage de $\mathbf{d}(i_{\boldsymbol{\mu}})$ est équivalent à encoder une unique sous-séquence. Ceci peut être fait efficacement à l'aide des techniques développées dans le chapitre 4. En particulier, nous utilisons la compression décrite à la section 4.4.2.[3] Le taux de compression utilisé ici est $r_{\boldsymbol{\mu}}$. La décompression demande une transformée en ondelettes inverse et permet d'obtenir $\hat{\mathbf{d}}(i_{\boldsymbol{\mu}})$.

Nous pourrions ensuite encoder $\mathbf{d}(i_1)$ de la même façon. Cependant, on sait que la sous-séquence $\mathbf{d}(i_{\boldsymbol{\mu}})$ lui est similaire et que nous pouvons l'utiliser comme une première approximation. Nous encodons donc plutôt la différence entre les sous-séquences, $\mathbf{q}(i_1) = \mathbf{d}(i_1) - \hat{\mathbf{d}}(i_{\boldsymbol{\mu}})$. Le vecteur $\mathbf{q}(i_1)$ peut être vu comme une sous-séquence au même titre que $\mathbf{d}(i_1)$, à la différence qu'il est beaucoup plus proche de zéro. On peut ainsi le compresser par ondelettes avec un taux de compression r_1 plus important que celui qui aurait dû être utilisé pour le signal original. La décompression se fait en appliquant tout d'abord une transformée en ondelettes inverse pour obtenir $\hat{\mathbf{q}}(j_1)$, puis en calculant $\hat{\mathbf{d}}(i_1) = \hat{\mathbf{q}}(j_1) + \hat{\mathbf{d}}(i_{\boldsymbol{\mu}})$.

On doit ensuite encoder le vecteur $\mathbf{v}(2)$. Celui-ci ne contient qu'un seul scalaire et il est stocké directement, sans compression. Par la suite, les autres vecteurs $\mathbf{v}(j)$ contiendront de plus en plus d'éléments, mais ceci reste une proportion négligeable de la taille totale de stockage requise. Nous n'avons ainsi pas cherché à les compresser.

Le stockage de $\mathbf{d}(i_2)$ se fait de façon similaire à celui de $\mathbf{d}(i_1)$. Ici, cependant, il est possible d'obtenir une approximation encore plus précise. On dispose en effet non seulement de $\hat{\mathbf{d}}(i_{\boldsymbol{\mu}})$ mais aussi de $\hat{\mathbf{d}}(i_1)$ et de $\mathbf{v}(2)$. On calcule ainsi la différence

$$\mathbf{q}(i_2) = \mathbf{d}(i_2) - \hat{\mathbf{d}}(i_{\boldsymbol{\mu}}) - \big[\hat{\mathbf{d}}(i_1) - \hat{\mathbf{d}}(i_{\boldsymbol{\mu}})\big]\mathbf{v}(2)$$

et le signal résultant est compressé par ondelettes en utilisant un taux de compression r_2.

[3] Nous n'utilisons pas la compression de la section 4.5 car celle-ci a été développée pour compresser des signaux $\mathbf{d}^{\text{D}}$ contenant directement des degrés de liberté alors qu'on est ici en présence des signaux $\mathbf{d}^{\text{I}}$ définis à la section 5.4.1.

Cette approche peut être généralisée et ainsi le stockage de $\mathbf{d}(i_j)$ se fait en encodant la différence suivante,

$$\mathbf{q}(i_j) = \mathbf{d}(i_j) - \hat{\mathbf{d}}(i_{\boldsymbol{\mu}}) - \left[\left(\hat{\mathbf{d}}(i_1) - \hat{\mathbf{d}}(i_{\boldsymbol{\mu}})\right) \cdots \left(\hat{\mathbf{d}}(i_{j-1}) - \hat{\mathbf{d}}(i_{\boldsymbol{\mu}})\right) \right] \mathbf{v}(j) \ ,$$

à l'aide d'une compression par ondelettes de taux r_j. Ici, la matrice à gauche de $\mathbf{v}(j)$ est une version approximative de E_{j-1}, ceci en raison des erreurs introduites lors de la compression. La décompression peut être réalisée facilement en appliquant tout d'abord une transformée en ondelettes inverse pour obtenir $\hat{\mathbf{q}}(i_j)$ puis en calculant

$$\hat{\mathbf{d}}(i_j) = \hat{\mathbf{q}}(i_j) + \hat{\mathbf{d}}(i_{\boldsymbol{\mu}}) + \left[\left(\hat{\mathbf{d}}(i_1) - \hat{\mathbf{d}}(i_{\boldsymbol{\mu}})\right) \cdots \left(\hat{\mathbf{d}}(i_{j-1}) - \hat{\mathbf{d}}(i_{\boldsymbol{\mu}})\right) \right] \mathbf{v}(j) \ .$$

Il faut noter qu'en général les approximations utilisées s'améliorent constamment avec j. Ceci nous permet d'augmenter graduellement le taux de compression r_j, ce qui constitue le principal bénéfice de la technique. En pratique, pour un groupe assez important, les dernières sous-séquences pourront être encodées à l'aide d'un très petit nombre de coefficients d'ondelettes. Il faut cependant spécifier précisément comment déteminer ce r_j.

À cette fin, la technique que nous proposons requiert que l'usager spécifie le taux de compression $r_{\boldsymbol{\mu}}$ qui sera utilisé pour encoder la première sous-séquence de chaque groupe. On utilise ce taux pour compresser $\mathbf{d}(i_{\boldsymbol{\mu}})$, puis on le décompresse pour obtenir la sous-séquence $\hat{\mathbf{d}}(i_{\boldsymbol{\mu}})$. L'erreur causée par la compression peut ensuite être évaluée par $\varepsilon^{\mathtt{M}} = |\mathbf{d}(i_{\boldsymbol{\mu}}) - \hat{\mathbf{d}}(i_{\boldsymbol{\mu}})|$. Ensuite, pour encoder $\mathbf{d}(i_j)$, on détermine simplement le r_j engendrant la même erreur de compression. Ceci peut se faire facilement en évaluant la somme du carré des coefficients d'ondelettes tronqués lors de la compression. Lorsque cette somme atteint $(\varepsilon^{\mathtt{M}})^2$, on sait que l'erreur engendrée par la compression de $\mathbf{d}(i_j)$ est égale à celle qui a été observée sur $\mathbf{d}(i_{\boldsymbol{\mu}})$.

6.3 Format de compression

La compression par *PCA* quantifiée peut être vue comme une compression par dictionnaire avec pertes. Cependant, à la différence des approches illustrées à la figure 6.1, les outils de correction associés à chaque entrée du dictionnaire, soient les vecteurs de base de la *PCA* quantifiée, représentent directement les données. Ainsi, le fait de décoder le dictionnaire fournit simultanément les données. Ce principe est illustré graphiquement à la figure 6.2. Nous décrivons maitenant le format de stockage de ce dictionnaire.

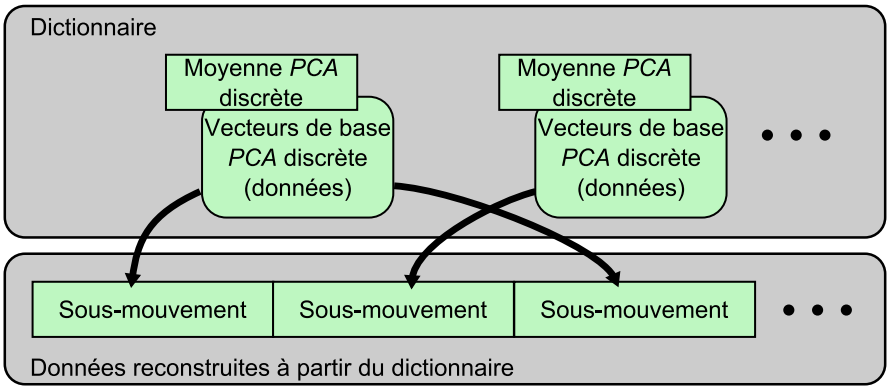


FIG. 6.2 – Compression par *PCA* quantifiée.

Le fichier compressé comprend une série de structures représentant chacune une entrée du dictionnaire. Chacune de ces structures est précédée d'un entier sur 32 bits indiquant le nombre d'octets qu'elle occupe.

Le premier élément composant une entrée de dictionnaire est une série de paires d'entiers. Ces entiers, encodés sur 16 bits, indiquent les indices de début et de fin des sous-séquences contenues dans cette entrée. Ils permettent ainsi de positionner précisément les sous-séquences à l'intérieur de la banque de données et d'en déterminer la longueur.

Suivent ensuite les coefficients d'ondelettes de $\hat{\mathbf{d}}(i_{\boldsymbol{\mu}})$, quantifiés sur 16 bits et encodés en utilisant l'approche décrite dans la section 4.7. Une approche similaire est ensuite utilisée pour quantifier les coefficients d'ondelettes de $\hat{\mathbf{q}}(i_1)$, mais la quantification s'effectue ici sur 8 bits étant donné que les valeurs minimales et maximales des coefficients sont beaucoup plus faibles. Le vecteur $\mathbf{v}(2)$, quant à lui, est stocké en format à virgule flottante 32 bits IEEE 754. Les coefficients d'ondelettes de $\hat{\mathbf{q}}(i_2)$ sont encodés de la même façon que ceux de $\hat{\mathbf{q}}(i_1)$. L'encodage se poursuit ainsi pour toutes les sous-séquences.

Le décodage se fait simplement en lisant séquentiellement les entrées du dictionnaire et en plaçant les sous-séquences obtenues au bon endroit dans la banque de données originale. Pour ce faire, il est nécessaire d'appliquer un rééchantillonnage étant donné que les sous-séquences compressées comptent toutes le même nombre d'échantillons, mais correspondent à des plages de longueurs différentes à l'intérieur de la banque de données.

ε^{P}	Ondelettes seules		*PCA* quantifiée	
0.70	25.8	(43:1)	22.6	(49:1)
0.44	32.5	(34:1)	28.9	(39:1)
0.25	49.3	(23:1)	44.5	(25:1)
0.15	68.2	(16:1)	62.1	(18:1)
0.03	235	(4.7:1)	216	(5.2:1)

(a) 13_29.amc

ε^{P}	Ondelettes seules		*PCA* quantifiée	
0.86	56.2	(23:1)	48.4	(27:1)
0.46	75.9	(17:1)	65.7	(20:1)
0.26	99.6	(13:1)	86.6	(15:1)
0.16	127	(10:1)	110	(12:1)
0.04	325	(4.0:1)	287	(4.6:1)

(b) 14_02.amc

TAB. 6.1 – Taille (en ko) et taux de compression de deux banques de données compressées de façon à obtenir une distorsion donnée ε^{P} (en cm). La première colonne montre le résultat lorsque chaque sous-séquence est uniquement encodée en ondelettes. La deuxième colonne montre le résultat de l'utilisation de la *PCA* quantifiée.

6.4 Résultats

Pour évaluer notre approche, nous avons testé la compression proposée sur deux des plus importantes banques de données évaluées à la section 5.8. La première, correspondant au fichier 13_29.amc dans la *CMU Motion Capture Database* [CMU], compte 4529 échantillons et contient différents mouvements d'exercices assez longs, chacun étant répété deux ou trois fois. La seconde, correspondant au fichier 14_02.amc, compte 5423 échantillons et représente un acteur s'adonnant à différents mouvements de boxe courts et fréquemment répétés.

En vue d'établir une base comparative, nous avons tout d'abord appliqué uniquement une compression en ondelettes de façon indépendante à chacune des sous-séquences. Le format de compression utilisé ici est très similaire à celui décrit à la section 6.3 à la différence qu'il n'est pas nécessaire de stocker les vecteurs $\mathbf{v}(j)$. De plus, tous les coefficients sont quantifiés sur 16 bits étant donné qu'on n'utilise pas la *PCA* quantifiée.

Le tableau 6.1 montre la taille des fichiers et les taux de compression obtenus pour différentes valeurs de distorsion ε^{P}. On remarque que l'utilisation de la *PCA* quantifiée permet d'augmenter le taux de compression par un facteur variant de 1.10 à 1.15. Cette augmentation peut paraître faible, mais l'amélioration due à la *PCA* quantifiée dépend en grande partie du

nombre et de la longueur des mouvements répétés dans la banque de données. Dans le cas de très vastes banques de données, il est à prévoir que ces mouvements redondants seront plus nombreux et ainsi augmenteront les bénéfices liés à l'utilisation de la *PCA* quantifiée.

Suite à une inspection visuelle nous avons remarqué que, pour une valeur de ε^{P} donnée, la qualité des animations compressées par *PCA* quantifiée est semblable à la qualité des animations compressées avec les différentes techniques présentées au chapitre 4, sauf aux jonctions entre deux sous-séquences. À ces endroits, en effet, la compression par *PCA* quantifiée introduit une discontinuité due au fait que le mouvement sortant est encodé indépendamment du mouvement entrant. Pour pallier à un problème similaire, Arikan [Ari06] a proposé une technique de fusion continue (*continuous merge*). Nous avons quant à nous opté pour une interpolation simple inspirée du *motion displacement mapping* de Bruderlin et Williams [BW95]. L'utilisation de cette interpolation réduit les artéfacts visibles lors d'une inspection visuelle mais engendre une légère augmentation de la distorsion positionnelle. Ceci tend donc encore une fois à montrer les limites de la métrique ε^{P} pour l'évaluation de la qualité visuelle. Les valeurs données au tableau 6.1 n'utilisent pas cette interpolation, mais nous montrons son effet dans la séquence vidéo accompagnant cette thèse [Bea07c].

Nous avons implanté la technique de référence et la compression par *PCA* quantifiée à l'aide de *Matlab*. Les test ont été exécutés sur un processeur Athlon64 3500+ muni de 2 GB de mémoire vive. Le temps de calcul requis pour la compression avec *PCA* quantifiée, comprenant l'extraction des vecteurs de base, la quantification et l'encodage, est de 1.66 s pour la banque de données 13_29.amc et de 4.95 s pour 14_02.amc. La décompression, quant à elle, s'effectue en 0.44 s pour 13_29.amc et 0.76 s pour 14_02.amc.

En vue de bien illustrer les gains issus de l'utilisation de la *PCA* quantifiée, nous avons évalué l'énergie résiduelle des signaux avant la compression en ondelettes. Cette énergie correspond en fait à la somme du carré des coefficients des différents $\mathbf{q}(i_1)$. La figure 6.3 illustre bien que l'énergie décroît progressivement avec l'augmentation du nombre de vecteurs de base.

6.5 Discussion

Dans cette section, nous présentons des limitations ou des extensions possibles reliées à la compression de groupes de mouvements.

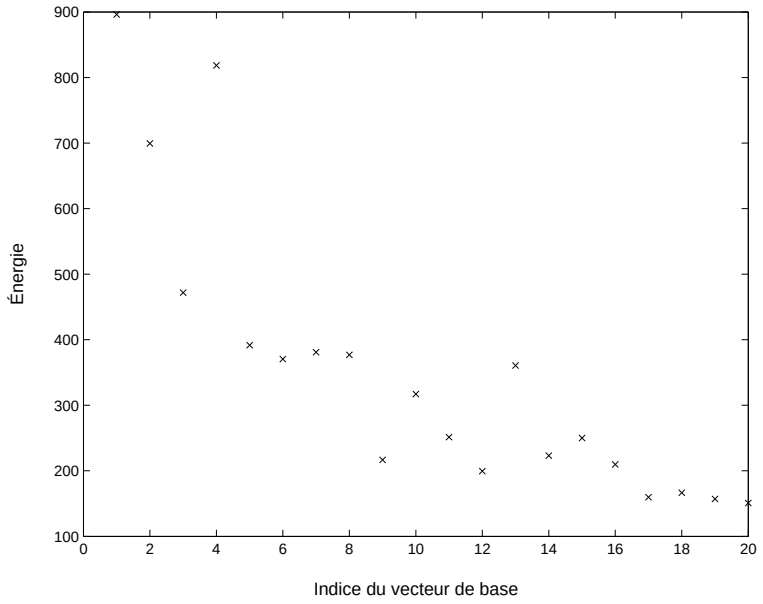


FIG. 6.3 – Ce graphe montre l'énergie résiduelle requise pour l'encodage des vecteurs de base successifs lors d'une compression par *PCA* quantifiée. Les vecteurs de base illustrés sont ceux du plus gros groupe de mouvements extrait de 14_02.amc.

6.5.1 Transmission continue

Un des problèmes de la méthode de compression proposée vient du fait que l'utilisation de la *PCA* quantifiée impose un ordre précis pour le décodage des sous-séquences d'un groupe. En général, cet ordre ne correspond pas à l'ordre d'apparition des sous-séquences dans la banque de données. Ainsi, si on souhaite décompresser la banque de données à un instant précis, il est possible qu'il s'avère nécessaire de décompresser des sous-séquences apparaissant beaucoup plus tard. En fait, on peut imaginer un cas pathologique où le décodage du premier échantillon de la banque de données requiert la décompression de toutes les sous-séquences qu'elle contient.

Cette situation ne cause pas de problème si on souhaite décompresser la totalité de la banque de données, mais elle rend l'utilisation de la *PCA* quantifiée inappropriée dans une application nécessitant une transmission continue des données (*data streaming*). En effet, dans ce genre d'application, les données doivent être reçues au fur et à mesure, décodées sur le champ et ainsi permettre un affichage séquentiel et graduel de la banque de données.

Pour pallier à ce problème, il serait possible de modifier l'algorithme de *PCA* quantifiée de façon à utiliser un nombre fixe de vecteurs de base pour chaque groupe de mouvements. Cette approche a l'inconvénient de produire une approximation moins précise des sous-séquences,

augmentant ainsi le coût de stockage des sous-séquences résiduelles. Cependant, elle serait en mesure de garantir que le décodage de la banque de données à n'importe quel instant ne nécessite, dans le pire cas, que la décompression d'un nombre constant de sous-séquences.

6.5.2 Continuité aux jonctions

Nous avons déjà mentionné que la compression indépendante des sous-séquences a l'inconvénient d'introduire des discontinuités lors de la transition d'un mouvement à un autre. Pour corriger ce problème, nous appliquons une correction *a posteriori* à l'aide d'une technique d'interpolation. Ce problème pourrait cependant être corrigé directement lors de la compression en s'assurant que, à chaque point de jonction, la sous-séquence sortante emprunte la même pose que la sous-séquence entrante.

À ce sujet, nous envisageons une approche où sont encodées, dans un premier temps, toutes les poses de jonction. Ces poses sont ensuite interpolées pour obtenir une approximation très grossière des animations composant la banque de données. Finalement les signaux correspondant aux sous-séquences sont déterminés en soustrayant les mouvements réels de l'approximation grossière.

Bien qu'une telle approche puisse être utilisée conjointement à une compression par *PCA* quantifiée, elle ne se prête pas à notre encodage par ondelettes. En effet, plusieurs des fonctions de la base d'ondelettes introduite à la section 4.3 sont non nulles au début et à la fin de l'intervalle. Ainsi, il est possible que la transformée en ondelettes tronquée introduise des déviations, et donc des discontinuités, aux points de jonction.

Ainsi, pour pallier à ce problème, il serait nécessaire d'utiliser un autre type de compression temporelle. Il pourrait par exemple être possible d'explorer l'utilisation du schéma de *lifting* pour la mise au point d'une base adaptée à des fonctions nulles au début et à la fin d'un intervalle. Alternativement, on pourrait explorer des techniques à base d'encodage par prédiction linéaire (*linear predictive coding*) [Say96].

Si besoin est, cette approche peut aussi être adaptée pour assurer non seulement la continuité des signaux, mais aussi celle de leurs premières dérivées.

6.5.3 Corrélation des degrés de liberté

La technique de compression présentée dans ce chapitre n'exploite pas spécifiquement les correlations existant entre les degrés de liberté des poses composant les sous-séquences. On

pourrait cependant tenter d'en tirer avantage en projetant par *PCA* l'ensemble de toutes les poses contenues dans un groupe de mouvements. Ceci pourrait en effet s'avérer plus efficace que l'application indépendante de la *PCA* à chaque mouvement (section 4.8.4) étant donné que, pour un groupe de mouvements, le coût de stockage des matrices de projection est amorti sur un plus grand nombre d'échantillons. Dans cette section, nous analysons plus en détail les avantages et les inconvénients d'une projection *PCA* des poses.

Lorsqu'on compresse les poses d'une animation par *PCA*, il est nécessaire de déterminer le nombre de dimensions qui seront conservées. À chaque dimension retenue correspond une colonne supplémentaire dans la matrice de projection ainsi qu'un signal de plus dans la version projetée des données. En supposant qu'on dispose de $k^{\mathtt{D}}$ degrés de liberté et de $n^{\mathtt{S}}$ échantillons, ceci implique donc l'ajout de $k^{\mathtt{D}} + n^{\mathtt{S}}$ scalaires.

De ces scalaires, les $n^{\mathtt{S}}$ éléments correspondant au nouveau canal de données projetées exhibent une cohérence temporelle permettant de les compresser efficacement par ondelettes. Par opposition, la compression que l'on peut appliquer à la matrice de projection obtenue par *PCA* se limite souvent à une quantification et un encodage entropique, deux opérations que l'on peut par ailleurs aussi appliquer aux coefficients d'ondelettes. À toute fin pratique, on peut donc ainsi dire que seuls les $n^{\mathtt{S}}$ nouveaux échantillons projetés seront compressés.

Pour simplifier l'analyse de l'effet de la *PCA* lors de la compression, nous supposons que tous les canaux temporels compressés en ondelettes conservent un même nombre de coefficients non nuls, soit $n^{\mathtt{S}}/r$. Ainsi, si la *PCA* n'est pas utilisée, on conservera un total de $k^{\mathtt{D}}n^{\mathtt{S}}/r$ scalaires. Si on choisit plutôt d'utiliser la *PCA* pour projeter sur $k^{\mathtt{PCA}}$ dimensions, alors le nombre total de scalaires à conserver est de $k^{\mathtt{D}}k^{\mathtt{PCA}} + k^{\mathtt{PCA}}n^{\mathtt{S}}/r$.

On peut donc conclure que l'utilisation de la *PCA* ne sera justifiée que si $k^{\mathtt{D}}k^{\mathtt{PCA}} + k^{\mathtt{PCA}}n^{\mathtt{S}}/r < k^{\mathtt{D}}n^{\mathtt{S}}/r$, soit si

$$n^{\mathtt{S}} > \frac{rk^{\mathtt{D}}k^{\mathtt{PCA}}}{k^{\mathtt{D}} - k^{\mathtt{PCA}}} \ .$$

Il faut donc un nombre minimum d'échantillons pour qu'il soit avantageux d'utiliser la projection *PCA* lors de la compression. La figure 6.4 illustre les valeurs de $n^{\mathtt{S}}$ minimales pour différentes valeurs de $k^{\mathtt{PCA}}$ et r, avec $k^{\mathtt{D}} = 62$. Par exemple, si pour un critère de qualité donné on détermine qu'il faut utiliser $r = 30$ et $k^{\mathtt{PCA}} = 20$, alors l'utilisation de la *PCA* ne sera justifiée que si on dispose d'au moins $n^{\mathtt{S}} = 885$ échantillons.

Un détail important omis par l'analyse précédente vient du fait que, pour un critère de qualité donné, le nombre de dimensions à conserver $k^{\mathtt{PCA}}$ a tendance à augmenter avec $n^{\mathtt{S}}$.

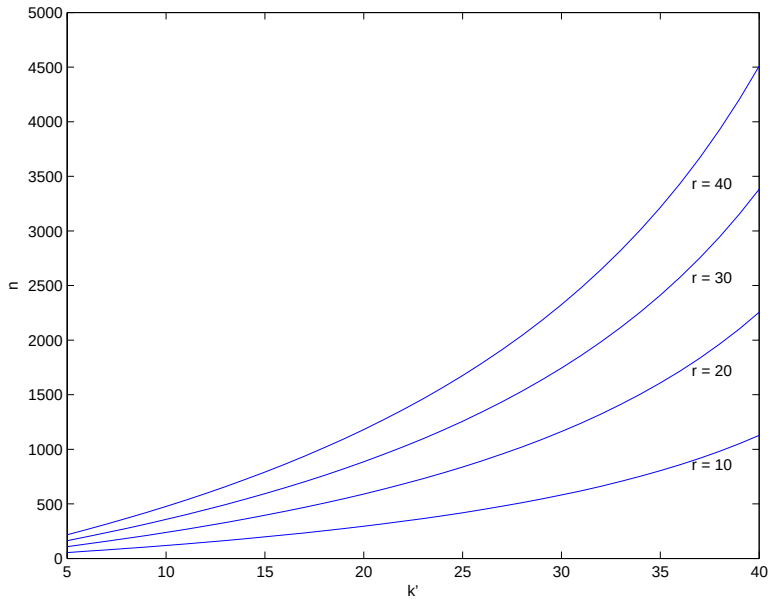


FIG. 6.4 – Valeurs minimales de $n^{\mathtt{S}}$, en fonction de $k^{\mathtt{PCA}}$ et r, justifiant l'utilisation d'une compression des poses par *PCA*. Ici, $k^{\mathtt{D}} = 62$.

Ceci vient du fait qu'en général, plus une animation compte d'échantillons, plus elle comporte de poses variées. Dans le cas de la compression introduite au chapitre 4, cette dépendance de $k^{\mathtt{PCA}}$ sur $n^{\mathtt{S}}$ fait en sorte qu'il est très difficile de trouver des situations où la *PCA* pourrait s'avérer efficace : pour un ratio donné r on est très souvent en-dessous de la courbe illustrée à la figure 6.4.

Cependant, dans le cas de la compression de groupes de mouvements, une même base *PCA* peut être utilisée pour projeter tous les mouvements du groupe. On peut ainsi considérer $n^{\mathtt{S}}$ non pas comme le nombre d'échantillons d'un mouvement mais comme le nombre total d'échantillons dans le groupe. Comme les mouvements du groupe sont très similaires, l'augmentation de $k^{\mathtt{PCA}}$ avec $n^{\mathtt{S}}$ se fait beaucoup plus lentement. Ceci vient du fait qu'une même base *PCA* peut représenter efficacement un très grand nombre d'échantillons différents. Il devient ainsi beaucoup plus facile d'envisager l'utilisation de la *PCA* dans le contexte de la compression présentée dans ce chapitre.

Les quelques expériences préliminaires que nous avons effectuées avec la *PCA* confirment les observations précédentes. Néanmoins, nous n'avons pas jugé nécessaire de développer formellement une technique de compression utilisant la *PCA*. La raison principale vient du fait que, parmi les groupes de mouvements que nous avons obtenus, très peu comptent un nombre

n^{S} d'échantillons assez élevé pour justifier l'utilisation de la *PCA*. On sait cependant que beaucoup de groupes assez gros pourraient être obtenus en appliquant la compression à de très grandes banques de données. Cependant, comme discuté à la section 5.9, certaines limitations techniques de notre algorithme d'identification de sous-séquences redondantes rendent difficile son application dans ce contexte.

6.5.4 Compression de mouvements synthétisés

Comme nous le verrons au chapitre suivant, plus spécifiquement aux sections 7.5.2 et 7.5.3, certaines applications peuvent être réalisées efficacement et simplement en enrichissant la banque de données par de nouveaux mouvements automatiquement synthétisés. Ces mouvements sont généralement obtenus à l'aide de combinaisons (*blending*) d'animations et peuvent donc s'encoder simplement en conservant les facteurs de pondération et les animations à combiner. Cependant, la technique de compression proposée dans ce chapitre fait en sorte que des mouvements issus de combinaisons seront en général encodés très efficacement. En effet, comme ces nouveaux mouvements s'expriment par une combinaison linéaire de sous-séquences déjà présentes dans le groupe, ils donneront lieu à un vecteur résiduel $\mathbf{q}(i_j)$ très près de zéro.

6.6 Conclusion

Dans ce chapitre, nous avons montré comment les compressions en ondelettes introduites au chapitre 4 peuvent être combinées à la technique de détection de mouvements redondants du chapitre 5 de façon à permettre la compression efficace d'une banque de données. Pour ce faire, nous avons mis au point une technique de compression par dictionnaire avec pertes. Nous avons ensuite montré différents problèmes d'une approche par *PCA* classique et avons introduit, en guise d'alternative, une technique originale nommée *PCA* quantifiée. La *PCA* quantifiée partage certaines caractéristiques de la *PCA* classique mais offre l'avantage que chaque vecteur de base encodé correspond précisément à une des données à compresser.

Suite à la présentation des résultats de compression obtenus sur deux banques de données comportant des répétitions de mouvements, nous avons discuté de différentes limitations ou extensions possibles à notre technique. En particulier, nous avons discuté de limitations quant à l'utilisation de cette compression pour la transmission continue de données (*data streaming*) et proposé des pistes de solutions. Nous avons aussi proposé des façons d'assurer la continuité

à la jonction entre les sous-séquences. Finalement, nous avons étudié les conditions permettant d'exploiter les correlations entre les degrés de liberté.

Dans le chapitre suivant, nous montrons comment les groupes de mouvements peuvent donner lieu à une structure de graphe et ainsi permettre la synthèse d'animations originales. Nous discutons aussi de différentes façons d'intégrer cette technique de synthèse à la compression de sous-séquences redondantes présentées dans ce chapitre.

Chapitre 7

Graphes compacts et synthèse d'animations

But you have not said any thing about this kingdom, path – whatever it is – behind the mirror. Did it answer your expectations ?

Colonel Grant

Nous avons vu que la segmentation en groupes de mouvements permettait d'interpréter une banque de données comme une séquence d'entrées à l'intérieur d'un dictionnaire. Cette interprétation s'est avérée fructueuse dans le cadre du développement d'une technique de compression, mais elle peut aussi être utilisée pour produire une représentation compacte de la banque de données sous la forme d'un graphe dirigé. En effet, on peut facilement associer un noeud à chaque groupe de mouvements puis les connecter suivant la séquence de mouvements présente dans la banque de données. Les graphes résultants sont très similaires aux graphes de mouvements proposés par d'autres auteurs et présentés à la section 3.3.5. Dans ce chapitre, nous montrons qu'ils peuvent aussi être utilisés pour synthétiser de nouvelles animations.[1]

Ce chapitre ne traite donc pas spécifiquement de compression, mais s'attarde plutôt à détailler une application connexe rendue possible par nos contributions précédentes. En ce sens, il se veut plutôt un survol rapide de techniques et de résultats qui risquent fort de faire l'objet de travaux de recherche futurs.

Nous présentons tout d'abord une définition formelle de la structure de graphe issue de la segmentation en groupes de mouvements. Par la suite, nous expliquons comment ces graphes peuvent être utilisés pour la synthèse d'animations et nous présentons certains résultats obtenus

[1] Dans ce chapitre, $\mathbf{d}$ désigne $\mathbf{d}^{\mathrm{I}}$.

à l'aide de cette technique. Finalement, nous concluons avec une discussion des avantages et d'extensions futures possibles.

7.1 Graphes de groupes de mouvements

Les sous-séquences similaires jouent un rôle important dans l'analyse des données. Elles représentent en effet des mouvements qui ont été répétés à de nombreuses reprises par l'acteur et qui correspondent ainsi souvent à des gestes sémantiquement significatifs, comme par exemple un saut ou un cycle de marche. Il semble naturellement intéressant d'organiser ces gestes sémantiques dans une structure de graphe indiquant les mouvements qui peuvent potentiellement se succéder.

En fait, une telle structure est pratiquement déjà contenue dans le résultat de l'algorithme présenté au chapitre 5. Les groupes de mouvements peuvent en effet être interprétés comme une façon de replier la banque de données sur elle-même de manière à ce que les sous-séquences similaires se superposent (figure 7.1). Cette interprétation conduit naturellement à une structure de graphe où chaque noeud représente un groupe de mouvements et chaque arc est une transition valide d'un groupe à un autre (figure 7.2(a)). Dans ce graphe, seuls les noeuds correspondant à des groupes de mouvements non dégénérés représentent des mouvements répétés par l'acteur. En effet, les autres noeuds contiennent exactement un arc entrant et un arc sortant.[2] Nous proposons donc de simplifier le graphe en omettant ces noeuds, de façon à illustrer plus clairement les transitions possibles entre les groupes non dégénérés.

Nous nommons le résultat, visible à la figure 7.2(b), graphe de groupes de mouvements. Celui-ci illustre très clairement les transitions possibles entre des noeuds correspondant souvent, comme nous l'avons expliqué plus haut, à des gestes sémantiquement significatifs. Une telle structure se rapproche des graphes de mouvements proposés par différents auteurs et que nous avons présentés en détail à la section 3.3.5. Comme nous le verrons dans la suite de ce chapitre, ces graphes peuvent être utilisés pour synthétiser de nouvelles animations en séquençant les mouvements contenus dans la banque de données.

[2]Le noeud "L", correspondant à un groupe dégénéré, déroge à cette règle et ne contient qu'un arc entrant car il est situé à la toute fin de l'animation. Il serait possible d'observer une exception similaire au début de l'animation.

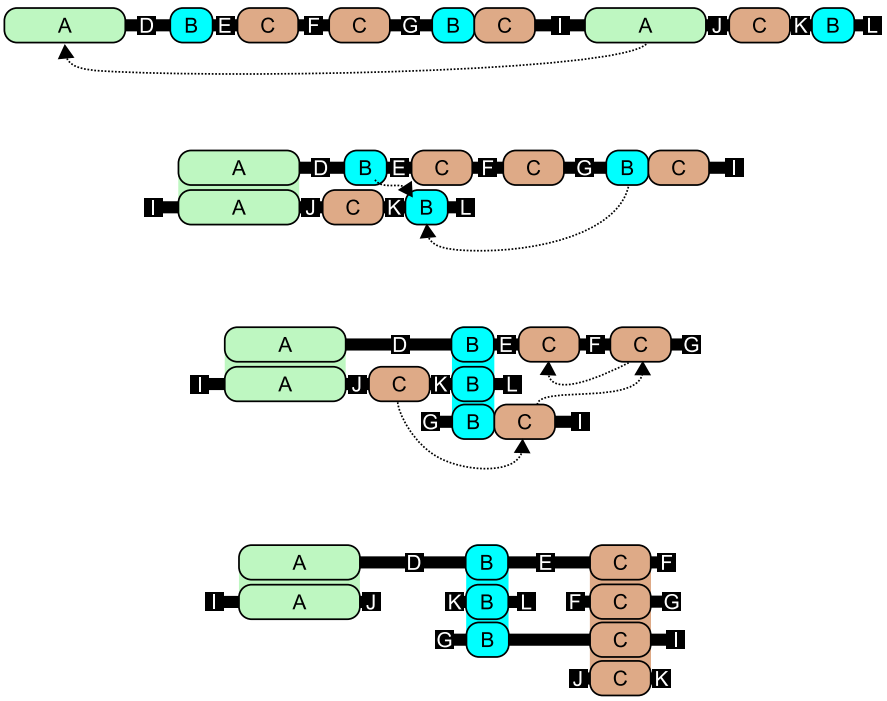
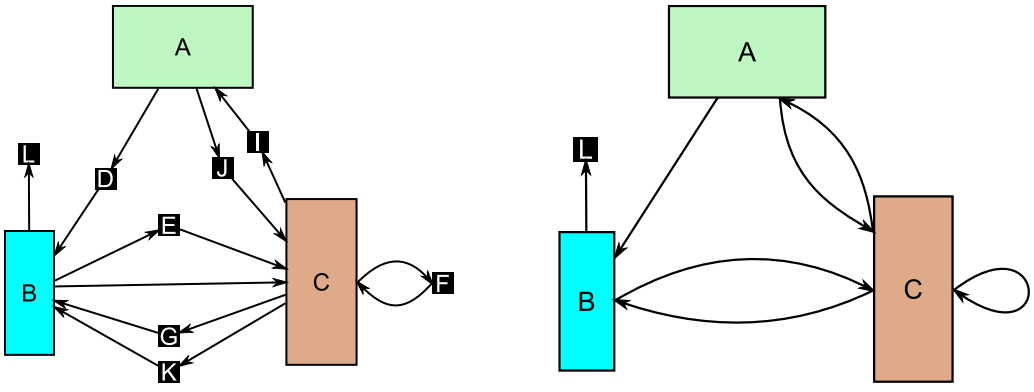


FIG. 7.1 – La segmentation en groupes de mouvements peut être interprétée comme une façon de replier la banque de données sur elle-même. Dans cette figure, A, B et C sont des groupes de mouvements non dégénérés tandis que les autres lettres représentent des groupes de mouvements dégénérés.



(a) Graphe direct (b) Graphe simplifié

FIG. 7.2 – Exemple du graphe de groupes de mouvements obtenu de la figure 7.1.

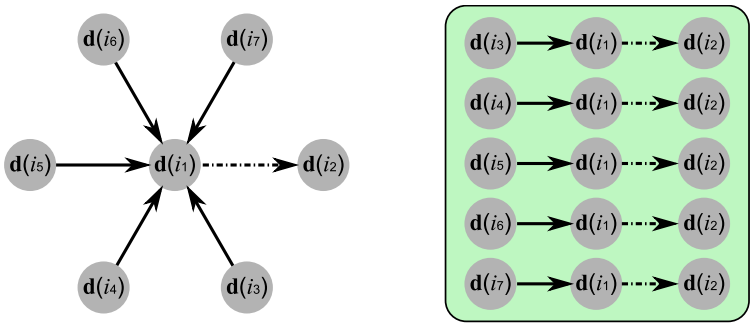


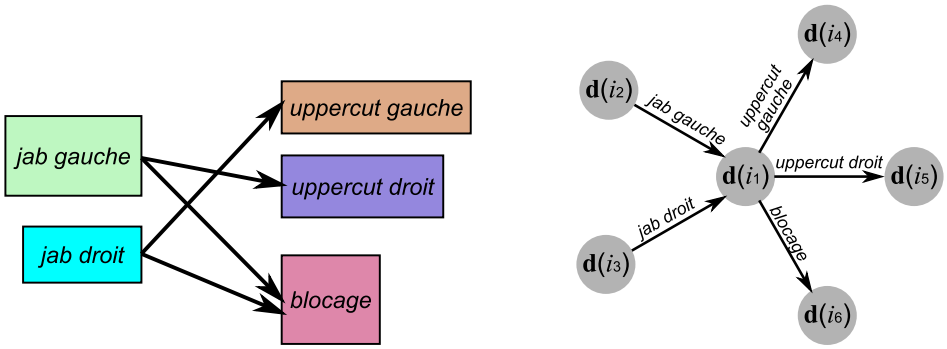
FIG. 7.3 – Dans un graphe noeud-pose, chaque mouvement correspond à un arc. Ainsi, l'ajout d'un unique arc peut impliquer la création d'un grand nombre de transitions. Dans cet exemple, l'ajout d'un mouvement entre les poses $\mathbf{d}(i_1)$ et $\mathbf{d}(i_2)$ crée cinq nouvelles transitions.

7.1.1 Graphes noeud-pose et noeud-mouvement

Dans la littérature sur les graphes de mouvements on distingue principalement deux types de graphes différents. Dans les premiers travaux sur le sujet [AF02, LCR+02, KGP02, GSKJ03], chaque noeud représente une pose et chaque arc correspond à un mouvement de transition entre deux poses. Cette approche, que nous nommons graphe noeud-pose, est aussi utilisée dans les *fat graphs* de Shin et Oh [SO06], à la différence que les arcs encodent une gamme de mouvements. Récemment, Heck et Gleicher [HG07] ont plutôt proposé un graphe noeud-mouvement où un noeud représente un mouvement paramétré et un arc indique une transition possible entre deux mouvements.

Bien que Heck et Gleicher ne le mentionnent pas explicitement, nous pouvons remarquer que le changement de structure qu'ils proposent offre certains avantages. En effet, dans les graphes noeud-pose, chaque mouvement entrant en un noeud doit pouvoir être suivi par tout mouvement sortant de ce noeud. Ceci implique que l'ajout d'un unique arc incident à un noeud peut nécessiter la création d'un grand nombre de transitions, tel qu'illustré à la figure 7.3. Cette explosion du nombre de transitions entre les mouvements peut rendre difficile la création de noeuds fortement connectés, utiles pour la synthèse interactive [GSKJ03].

Par opposition, les arcs de la structure proposée par Heck et Gleicher [HG07] indiquent directement des transitions entre les mouvements. L'ajout d'un arc correspond donc à l'ajout d'une unique transition. La figure 7.4 montre aussi un exemple simple où les transitions encodées dans un graphe noeud-mouvement ne peuvent pas être reproduites par un graphe noeud-pose.



(a) Graphe noeud-mouvement (b) Graphe noeud-pose

FIG. 7.4 – Comparaison entre un graphe noeud-pose et un graphe noeud-mouvement. Le graphe (b) semble équivalent au graphe (a) mais il force l'ajout des transitions non désirées *jab gauche* → *uppercut gauche* et *jab droit* → *uppercut droit*.

L'inconvénient principal des graphes noeud-mouvement, comme nous l'avons illustré à la figure 7.2(a), vient du fait qu'ils peuvent contenir un grand nombre de noeuds ne possédant qu'un arc entrant et un arc sortant. Ces noeuds n'offrent pas de possibilité de branchement et s'avèrent donc superflus lors d'une recherche dans le graphe. Nous proposons donc la structure hybride illustrée à la figure 7.2(b) où certains arcs représentent des transitions directes entre deux noeuds et d'autres sont associés à un mouvement transitoire.

7.1.2 Définition de la structure

Nous définissons maintenant formellement la structure de graphe de groupes de mouvements. Rappelons que l'algorithme d'identification des sous-séquences redondantes produit un ensemble de groupes de mouvements,

$$\left\{M(1), M(2), ...\right\} .$$

Chacun de ces groupes correspond à son tour à un ensemble de sous-séquences,

$$M = \left\{\mathbf{d}[i_1, j_1], \mathbf{d}[i_2, j_2], ...\right\} .$$

Rappelons que la hauteur du groupe, $|M|$, correspond au nombre de sous-séquences dans ce groupe. La largeur du groupe, $\tilde{M}$, est plutôt le nombre moyen d'échantillons dans les sous-séquences et le volume est définit comme suit,

$$\text{vol}(M) = (|M| - 1)\tilde{M}^{\rho} ,$$

où ρ est un paramètre contrôlé par l'usager et introduit à la section 5.5.3.

On souhaite associer un noeud à chaque groupe de mouvements non dégénéré. On définit dont l'ensemble des noeuds de la façon suivante,

$$N = \left\{ \; j \; \middle| \; |M(j)| > 1 \; \right\} .$$

Pour définir les arcs, nous introduisons tout d'abord des ensembles permettant d'identifier si deux groupes de mouvements sont adjacents. L'ensemble suivant,

$$T^{\texttt{DIR}}(o, o') = \left\{ (i_1, j_1, i_2, j_2) \; \middle| \; \mathbf{d}[i_1, j_1] \in M(o) \; , \; \mathbf{d}[i_2, j_2] \in M(o') \; , \; j_1 = i_2 \right\} \; ,$$

sera non vide seulement si les groupes $M(o)$ et $M(o')$ sont contigus dans la banque de données. Pour identifier aussi les groupes séparés par un groupe dégénéré on définit

$$T^{\texttt{IND}}(o, o') = \left\{ (i_1, j_1, i_2, j_2) \; \middle| \; \begin{array}{l} \mathbf{d}[i_1, j_1] \in M(o) \; , \; \mathbf{d}[i_2, j_2] \in M(o') \; , \\ \exists o'' \;\; \text{t.q.} \;\; \mathbf{d}[j_1, i_2] \in M(o'') \;\; \text{et} \;\; |M(o'')| = 1 \end{array} \right\} \; .$$

On peut ensuite faire l'union de ces deux ensembles, $T(o, o') = T^{\texttt{DIR}}(o, o') \cup T^{\texttt{IND}}(o, o')$. Ceci nous permet finalement de définir les arcs du graphe comme un ensemble de tuples,

$$A = \left\{ \; (o, o') \; \middle| \; T(o, o') \neq \emptyset \; \right\} \; . \tag{7.1}$$

7.2 Création de graphes

Il est assez facile de concevoir un algorithme permettant de générer les différents ensembles précédents à partir des groupes de mouvements. Pour construire l'ensemble N, il suffit simplement d'itérer à travers les groupes pour identifier ceux qui ne sont pas dégénérés. Pour les ensembles $T(o, o')$ et A, on parcourt plutôt toutes les sous-séquences dans l'ordre dans lequel elles apparaissent dans la banque de données. À chaque fois que l'on rencontre une sous-séquence $\mathbf{d}[i_1, j_1]$ membre d'un groupe non dégénéré $M(o)$, on identifie la prochaine sous-séquence $\mathbf{d}[i_2, j_2]$ membre d'un groupe non dégénéré $M(o')$. Les indices de début et de fin des deux sous-séquences permettent de former le 4-tuple (i_1, j_1, i_2, j_2) qui est ensuite ajouté à l'ensemble $T(o, o')$. Finalement, le 2-tuple (o, o') est ajouté à A. Notez que, si deux groupes non dégénérés sont adjacents, alors $j_1 = i_2$.

Le graphe défini par les ensembles N et A peut être illustré facilement en dessinant les noeuds et les arcs. Nous avons choisi d'utiliser des boîtes rectangulaires pour illustrer les noeuds. Ceci nous permet d'ajuster la taille de la boîte de manière à refléter la nature du groupe de mouvements associé. Nous dessinons donc des boîtes de hauteur proportionelle à $(|M| - 1)$

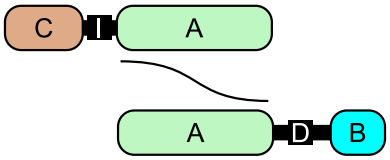


FIG. 7.5 – Cette figure illustre la façon de synthétiser une animation originale à partir d'un graphe de groupes de mouvements. Les données utilisées dans cet exemple sont tirées de la figure 7.1, résultant dans le graphe de la figure 7.2(b). Dans ce graphe, la présence d'un arc entre les groupes de mouvements C et A implique l'existence, dans les données, d'une transition entre des mouvements de ces deux groupes, illustrée dans le haut de la figure. Même chose pour la transition entre les groupes A et B, illustrée dans le bas. Le mouvement correspondant à l'enchaînement C $\rightarrow$ A $\rightarrow$ B n'existe pas dans les données mais peut être synthétisé en combinant les deux mouvements du groupe A. Cette combinaison est indiquée par la courbe au centre de la figure.

et de largeur proportionelle à $\tilde{M}^{\rho}$. Ainsi, l'aire de la boîte est proportionnelle au volume du groupe. Un graphe du genre est illustré à la figure 7.2(b).

7.3 Synthèse d'animations

Le graphe obtenu encode donc toutes les transitions de mouvements qui ont été observées dans la banque de données. Il ne donne cependant aucune façon de produire un enchaînement de mouvements différent de l'unique séquence enregistrée lors de la séance de capture de mouvements. Dans ce chapitre, nous proposons une méthode simple qui exploite les caractéristiques des groupes de mouvements pour permettre de synthétiser n'importe quel enchaînement de mouvements permis par le graphe. Le principe directeur est simple : partant du fait que toutes les sous-séquences présentes dans un groupe de mouvements sont similaires, nous nous permettons de combiner (*blending*) n'importe quelles de ces sous-séquences.

La technique est illustrée à la figure 7.5. Plus formellement, imaginons qu'on souhaite synthétiser une animation correspondant à un parcours du graphe quelconque débutant au noeud o_1 et se terminant à o_k. Le parcours suit donc l'enchaînement de noeuds suivant :

$$o_1 \rightarrow o_2 \rightarrow o_3 \rightarrow ... \rightarrow o_k \ .$$

Si on s'intéresse à l'arc $o_1 \rightarrow o_2$, alors l'équation 7.1 nous garantit que $T(o_1, o_2) \neq \emptyset$. Soit

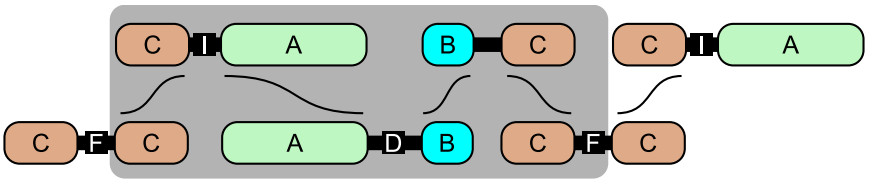


FIG. 7.6 – Un graphe de groupes de mouvements contenant un cycle peut être utilisé pour synthétiser une animation cyclique. Dans cet exemple, l'animation synthétisée dans l'encadré gris est cyclique.

(i_1, j_1, i_2, j_2) un 4-tuple choisi au hasard dans $T(o_1, o_2)$. Ce 4-tuple nous indique que $\mathbf{d}[i_1, j_1]$ est une sous-séquence contenue dans $M(o_1)$, que $\mathbf{d}[j_1, i_2]$ (si $j_1 \neq i_2$) est membre d'un groupe dégénéré agissant comme une transition, et que $\mathbf{d}[i_2, j_2]$ est une sous-séquence contenue dans $M(o_2)$. On peut ainsi produire les premiers échantillons de l'animation synthétisée :

$$(i_1, i_1 + 1, ..., j_1, ..., i_2 - 1) \ .$$

Pour générer les échantillons suivants, nous devons choisir un autre 4-tuple, cette fois dans l'ensemble $T(o_2, o_3)$. Encore une fois, l'existence d'un tel 4-tuple est garantie par la présence de l'arc $o_2 \rightarrow o_3$ et par l'équation 7.1. Soit (i'_2, j'_2, i_3, j_3) le 4-tuple choisi de la sorte. Ici, on sait que la sous-séquence $\mathbf{d}[i'_2, j'_2]$ est similaire à $\mathbf{d}[i_2, j_2]$ étant donné qu'elles sont toutes deux dans le groupe de mouvements $M(o_2)$.

Les échantillons suivants de l'animation peuvent alors être synthétisés en combinant les sous-séquences $\mathbf{d}[i_2, j_2 - 1]$ et $\mathbf{d}[i'_2, j'_2 - 1]$. Le facteur de combinaison varie lentement de 1 à 0, de façon à ce que le premier échantillon produit soit i_2 et le dernier soit j'_2. Nous avons fait varier ce facteur à l'aide d'une fonction cubique dont la dérivée est nulle au début et à la fin.

Remarquons que, bien qu'elles soient similaires, les sous-séquences $\mathbf{d}[i_2, j_2 - 1]$ et $\mathbf{d}[i'_2, j'_2 - 1]$ ne comptent pas nécessairement exactement le même nombre d'échantillons. Pour corriger ce problème, nous synthétisons un nombre d'échantillons égal à la moyenne de la longueur de ces deux sous-séquences. De plus, nous rééchantillonnons uniformément les deux sous-séquences pour les faire correspondre aux échantillons synthétisés.

Suite à l'échantillon $j'_2 - 1$, nous synthétisons la séquence d'échantillons $(j'_2, j'_2 + 1, ..., i_3 - 1)$, qui correspond à un groupe de mouvements dégénérés agissant comme une transition. On poursuit ensuite en identifiant $(i'_3, j'_3, i_4, j_4) \in T(o_3, o_4)$ et en combinant $\mathbf{d}[i_3, j_3 - 1]$ et $\mathbf{d}[i'_3, j'_3 - 1]$, suivi de la transition $(j'_3, j'_3 + 1, ..., i_4 - 1)$ et ainsi de suite.

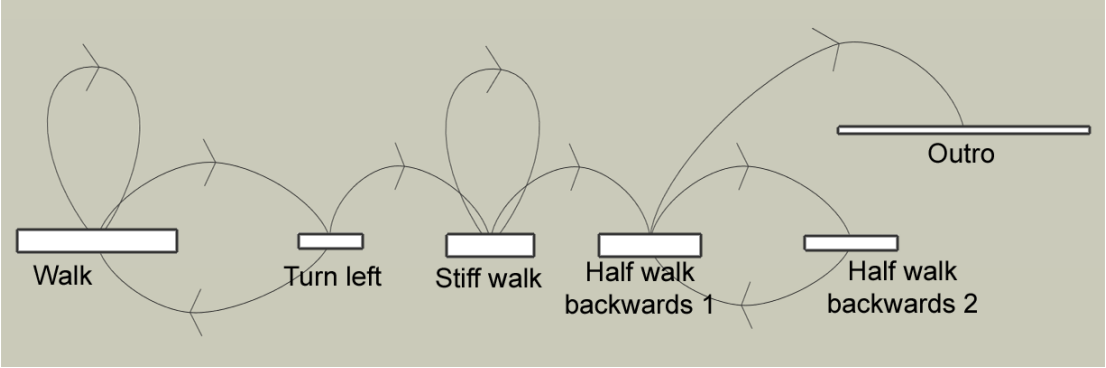


FIG. 7.7 – Graphe de groupes de mouvements pour une animation de marche.

Cette approche peut être utilisée pour créer des animations ayant un début et une fin mais, si le graphe de groupes de mouvements contient un cycle, alors il est aussi possible de générer une animation cyclique, telle qu'illustrée à la figure 7.6.

7.4 Résultats

Nous avons utilisé l'algorithme et la structure de graphe présentés dans ce chapitre sur les différentes banques de données mentionnées à la section 5.8. Dans tous les cas, la création du graphe requiert moins de 15 ms. Les graphes obtenus contiennent tous les mouvements de la banque de données. Une partie de ces mouvements sont inclus dans les noeuds, sous forme de groupes de mouvements non dégénérés, tandis que le reste de la banque de données, correspondant à des groupes dégénérés, est encodé dans des arcs.

La première question qui nous intéresse est la suivante : "Est-ce que l'algorithme proposé permet d'obtenir une structure de graphe raisonnable à partir de données non structurées ?" Bien que cette question demeure assez vague, nous présentons un certain nombre de résultats constituant des éléments de réponse.

La figure 7.7 montre le graphe de groupes de mouvements obtenu pour la banque de données de marche (1900 échantillons à 120 Hz). Nous avons disposé et annoté les noeuds manuellement de façon à mettre en évidence la structure et le contenu des différents noeuds. Cinq noeuds principaux ont été identifiés, chacun contenant des mouvements très similaires.

La figure 7.8 montre le graphe de groupes de mouvements obtenu pour la banque de données de boxe (5400 échantillons à 120 Hz). Ici encore la disposition et l'annotation ont été faites manuellement. L'algorithme d'extraction de groupes de mouvements utilise quant à lui un pa-

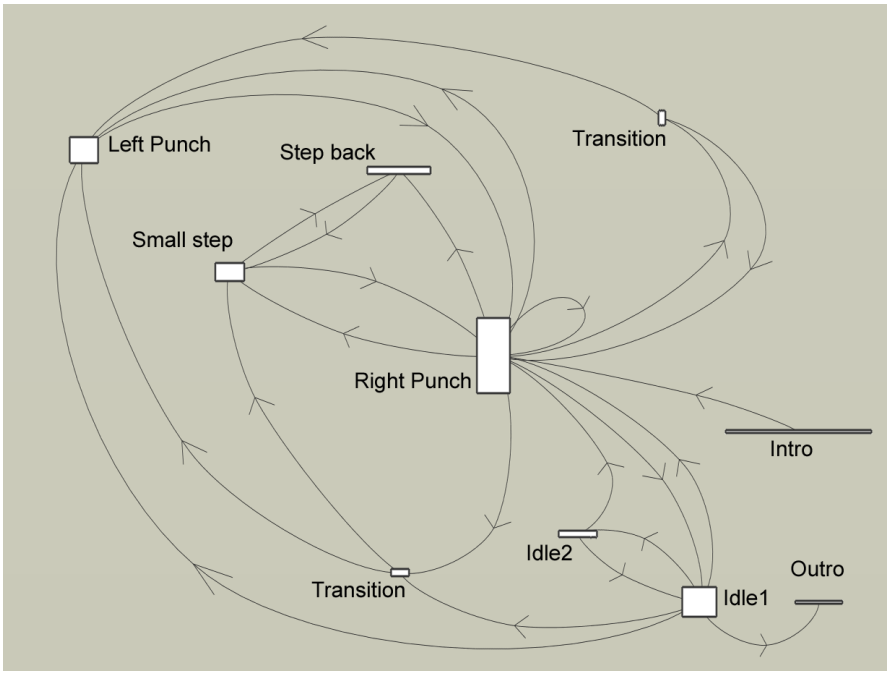


FIG. 7.8 – Graphe de groupes de mouvements pour une animation de boxe avec $\rho = 1$.

ramètre $\rho = 1$ (section 5.5.3). Le contenu des noeuds correspondant aux coups des poings droits et gauches sont illustrés à la figure 5.14.

La figure 7.9 montre le graphe obtenu à partir de la même banque de données, mais en utilisant cette fois $\rho = 2$, favorisant du même coup les groupes de mouvements plus longs. Ceci permet l'apparition de plus de noeuds, correspondant à des types de coups de poing différents. On remarque aussi un nouveau noeud contenant des combinaisons de coups de poing droite-gauche.

Ce graphe illustre aussi les trous dans la connectivité. Par exemple, il n'existe aucune transition permettant de passer directement d'une combinaison droite-gauche à un *uppercut*. Ceci peut correspondre à des contraintes logiques souhaitées par l'utilisateur. Il est aussi possible que ces transitions soient désirées mais qu'elles n'aient pas été exécutées par l'acteur lors de la séance de capture. À la section 7.5.2, nous proposons un méthode qui pourrait permettre de contourner ce problème.

Pour nous assurer de la robustesse de l'algorithme, nous l'avons exécuté sur la juxtaposition des banques de données de marche et de boxe. Tel qu'attendu, le graphe résultant, illustré à la figure 7.10, exhibe deux structures disjointes très similiaires à celles des figures 7.7 et 7.8. Les différences sont dues aux variations naturelles de l'algorithme présenté au chapitre 5, qui

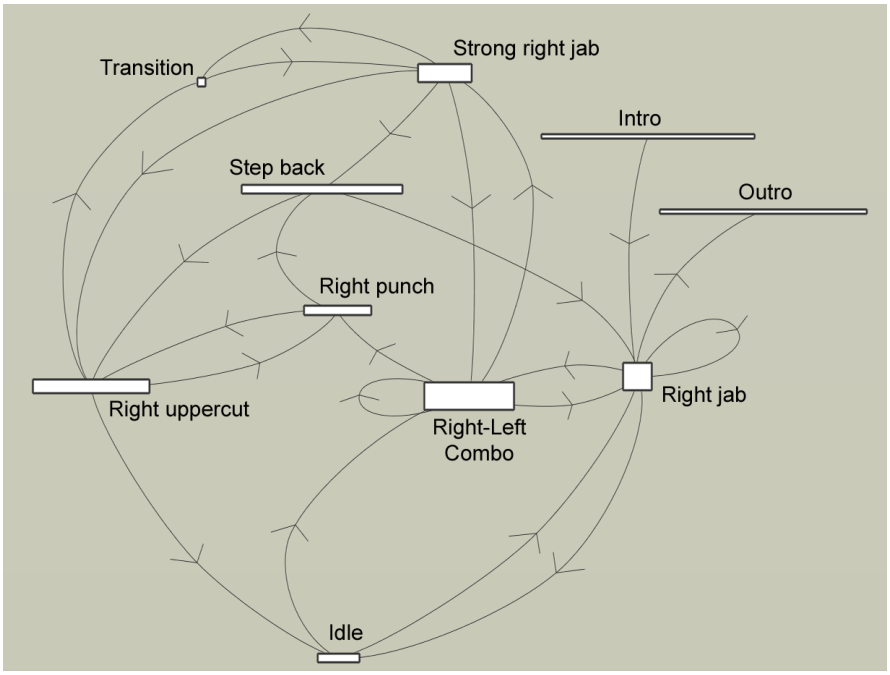


FIG. 7.9 – Graphe de groupes de mouvements pour une animation de boxe avec $\rho = 2$.

résultent à la fois de l'agrégation des poses et des choix aléatoires faits lors de l'étape vorace.

Nous avons aussi testé le résultat de l'algorithme d'extraction de graphe sur des données bruitées, en l'occurence les animations de boxe où, pour chaque pose, chaque degré de liberté est perturbé par un bruit gaussien indépendant de moyenne nulle et de variance 0.1. Comme nous l'avions déjà mentionné à la section 5.8, l'ajout de bruit a peu d'impact sur le résultat, visible ici à la figure 7.11.

7.4.1 Synthèse de mouvements

Nous avons utilisé les graphes présentés précédemment pour synthétiser de nouvelles animations. Chacune de ces animations, non cycliques, a été produite en déterminant à l'avance un enchaînement aléatoire de noeuds connectés par des arcs. Ces animations sont visibles dans le vidéo associé à cette thèse [Bea07b]. La figure 7.12 montre quelques poses tirées d'une telle animation. Nous avons aussi utilisé une courte séquence contenant une animation de course non cyclique pour générer une animation de course cyclique.

Les animations présentées sont synthétisées en temps réel. Les temps d'exécution de l'algorithme de synthèse n'ont pas été précisément évalués, mais celui-ci est très rapide car il ne requiert que le décodage de deux poses et leur combinaison (*blending*).

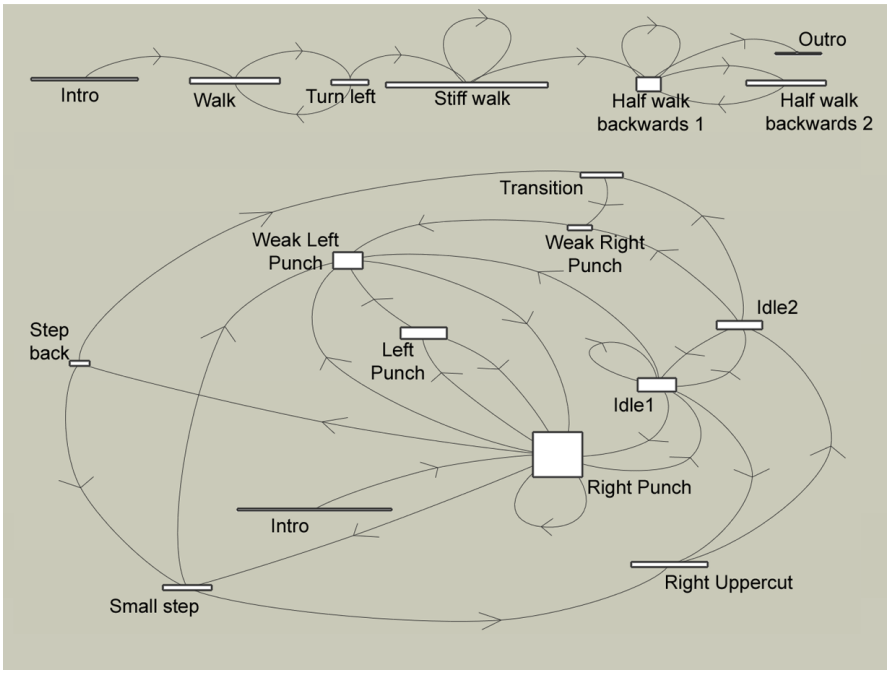


FIG. 7.10 – Graphe de groupes de mouvements obtenu pour l'union de l'animation de marche et de l'animation de boxe avec $\rho = 1$.

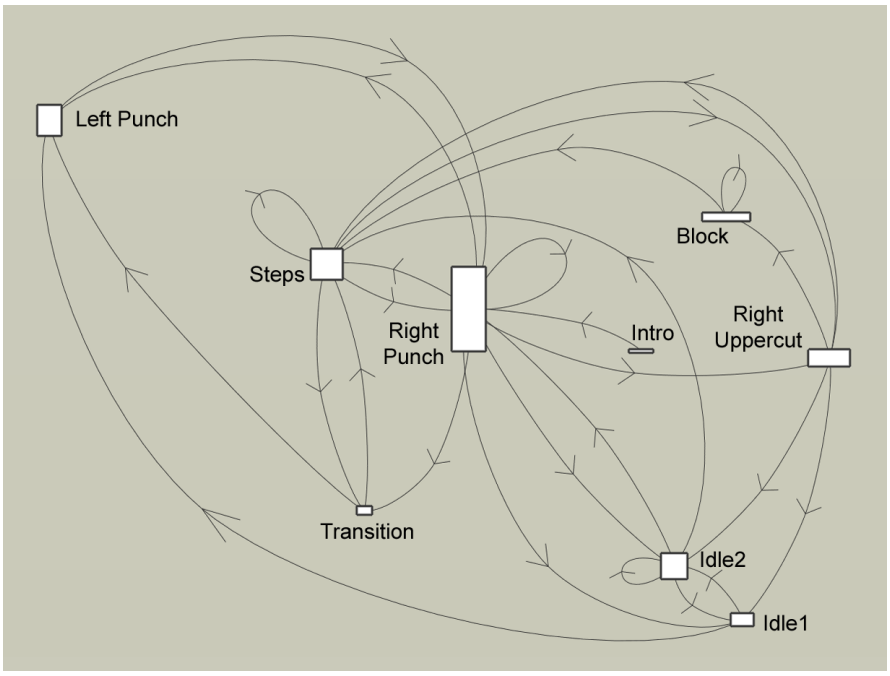


FIG. 7.11 – Graphe de groupes de mouvements pour une animation de boxe bruitée avec $\rho = 1$.

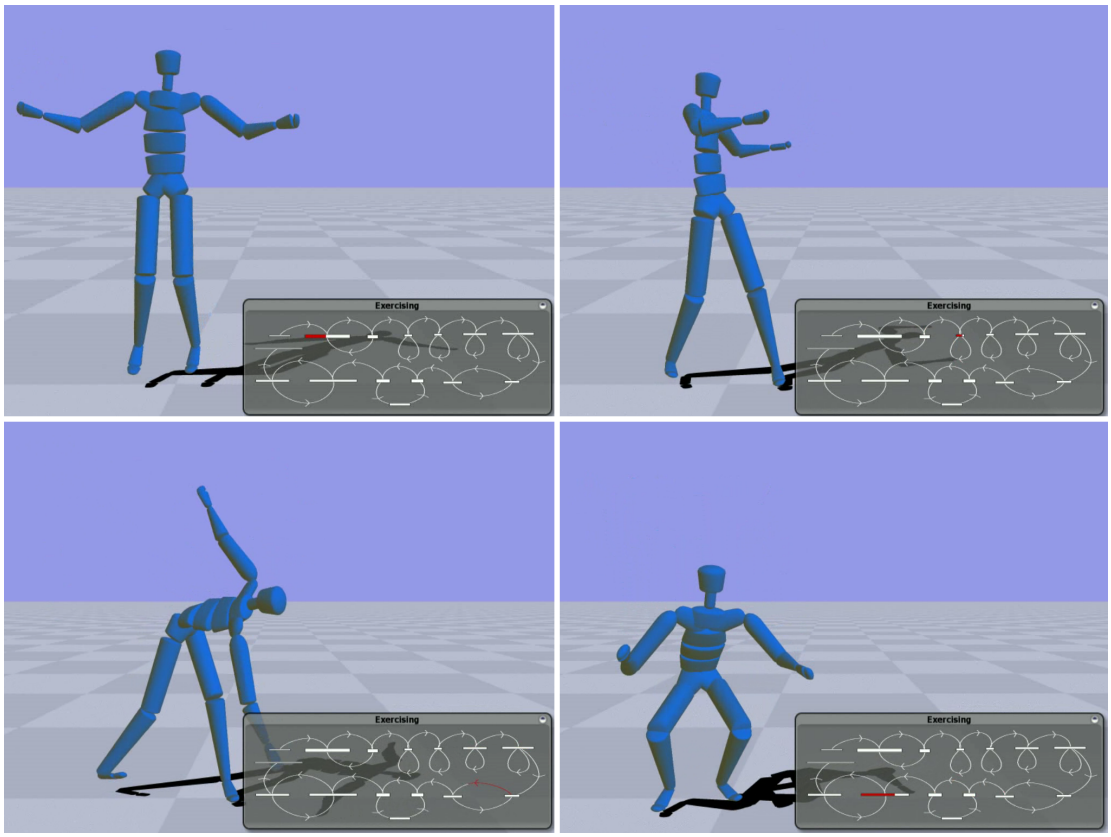


FIG. 7.12 – Poses tirées d'une animation synthétisée en utilisant un graphe de groupes de mouvements.

7.5 Discussion

La création de graphes compacts et leur utilisation pour la synthèse d'animations est le dernier volet des recherches qui ont été explorées dans le cadre de ce doctorat. Elles se sont néanmoins révélés forts prometteuses et font donc naturellement partie de travaux dont l'évolution se poursuit à ce jour. Dans la suite de cette section nous présentons différentes diections que nous envisageons explorer. Certains des aspects sont abordés de façon détaillés, reflétant l'avancement actuel de nos travaux. D'autres sections sont plutôt exploratoires et témoignent d'aspects que nous souhaitons aborder dans le futur.

Certains des aspects abordés ici illustrent bien les liens entre la compression de groupes de mouvements, présentée au chapitre précédent, et leur utilisation pour la synthèse d'animations.

7.5.1 Correction lors de la combinaison

Comme nous l'avons déjà vu, la synthèse d'animations que nous proposons requiert la combinaison (*blending*) de deux mouvements. Bien que ces mouvements soient similaires et qu'ils soient bien alignés au début et à la fin, il est possible que leurs poses intermédiaires ne correspondent pas précisément. De plus, il est possible que les contraintes de contact diffèrent. Par exemple, dans le premier mouvement le pied pourrait être posé au sol plus longtemps que dans le second mouvement avec lequel il est combiné.

Différentes stratégies peuvent être utilisées pour résoudre ces problèmes. Parmi celles-ci, on trouve le *dynamic time-warping* [BW95], les courbes d'enregistrement [KG03] et la correction par cinématique inverse (voir section 4.6). Comme le comportement de ces techniques est déjà bien connu, nous avons choisi de ne pas les étudier dans le contexte de la synthèse par graphe de groupes de mouvements. Ainsi, les animations accompagnant cette thèse ne font pas appel à ces différentes stratégies.

Ceci nous permet de mieux observer la qualité brute de notre système de synthèse, en plus d'illustrer le fait que les animations contenues dans un groupe de mouvements peuvent être combinées efficacement. En contrepartie, de brefs glissements de pieds et de légers artéfacts de synthèse peuvent parfois être observés. En pratique, une implantation complète de la technique devrait s'assurer de corriger ces problèmes.

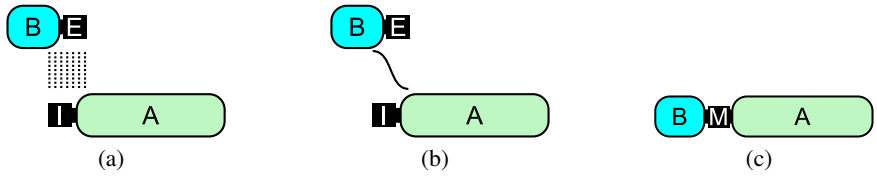


FIG. 7.13 – Synthèse automatique de nouvelles transitions entre deux groupes de mouvements. (a) On identifie tout d'abord un mouvement du premier groupe dont les dernières poses sont similaires aux premières poses d'un mouvement du second groupe. (b) Un mécanisme d'ajustement est ensuite utilisé pour produire une transition sans coupure. (c) Finalement, le nouvel enchaînement de mouvements résultant est ajouté à la banque de données compressée.

7.5.2 Synthèse de transitions

Il est possible que la fin d'un mouvement soit très similaire au début d'un autre mouvement mais que les noeuds correspondants ne soient pas reliés par un arc dans le graphe de groupes de mouvements. Ceci indique que, malgré leur similarité, l'acteur n'a jamais explicitement effectué une transition entre ces mouvements. Dans certains cas, ce comportement peut être désirable, par exemple si des contraintes logiques de plus haut niveau empêchent ces mouvements de se succéder. Nous avons d'ailleurs mentionné cet avantage à la section 7.1.1.

Dans d'autres cas, cependant, cette caractéristique des graphes de groupes de mouvements fait en sorte que l'acteur devra effectuer un grand nombre de transitions explicites alors même que celles-ci pourraient être facilement synthétisées. Pour pallier à ce problème, il serait possible d'enrichir la banque de données avec des transitions synthétisées.

Imaginons par exemple que l'utilisateur souhaite ajouter un arc entre les groupes de mouvements B et A sans avoir à capturer d'animations supplémentaires. Ceci pourrait être fait en choisissant, dans le groupe B, une animation dont les dernières poses sont proches des premières poses d'une autre animation tirée du groupe A (figure 7.13(a)). De nouveaux échantillons peuvent ensuite être ajoutés aux données en juxtaposant ces deux mouvements et en les ajustant de façon à produire une transition sans rupture. À cet effet, les techniques d'ajustement d'animations proposées par Gleicher *et al.* [GSKJ03] pourraient être utilisées (figure 7.13(b)).

Pour éviter que les animations synthétisées n'augmentent la taille de stockage de la banque de données il serait possible de complexifier les structures de données de façon à permettre la création automatique de ces transitions lors de la synthèse. Cependant, nous croyons que ces nouveaux mouvements pourraient être stockés à très faible coût en utilisant la technique de

compression proposée au chapitre 6 (figure 7.13(c)).

Bien que nous n'ayons pas implanté spécifiquement cette approche, la discussion précédente nous porte à penser que les graphes de groupes de mouvements peuvent être considérés comme une généralisation des graphes de mouvements usuels [KGP02].

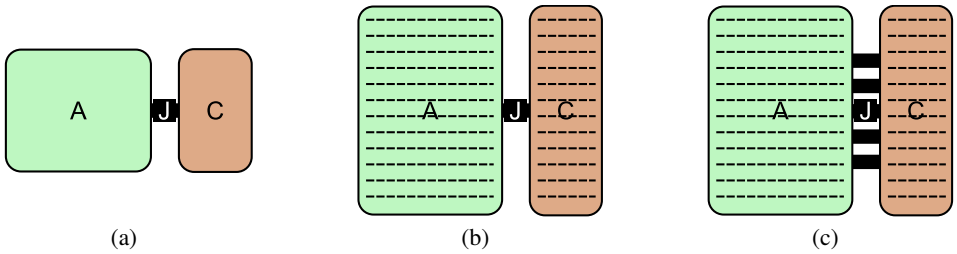
7.5.3 Graphes paramétriques

Nous avons vu que les mouvements contenus dans un noeud peuvent être combinés (*blending*) de façon à synthétiser un nouvel enchaînement de mouvements. Cependant, il est aussi possible de combiner ces mouvements dans le but de produire une gamme continue de mouvements originaux. Par exemple, un saut court et un saut long appartenant au même groupe de mouvements pourraient être combinés pour créer des sauts pour toutes les distances intermédiaires. En ce sens, les graphes de groupes de mouvements peuvent être comparés aux *fat graphs* [SO06] et aux graphes paramétriques [HG07], présentés en détail à la section 3.3.5.

Ceci soulève cependant le problème de la paramétrisation. En effet, la relation entre la pondération utilisée lors de la combinaison et une notion concrète, comme la longueur du saut dans l'exemple précédent, est en général non-linéaire et ne peut être facilement déterminée. Pour ce faire, nous pouvons utiliser la technique d'échantillonnage de combinaisons proposée par Kovar et Gleicher [KG04b] (figure 7.14(b)). Encore une fois, ici, il serait possible de complexifier la structure de graphe pour y encoder la relation entre la pondération des combinaisons et la valeur des paramètres concrets envisagés. Cependant, nous proposons plutôt d'ajouter à la banque de données les nouveaux mouvements issus de l'échantillonnage et de nous fier sur la compression introduite au chapitre 6 pour éviter d'accroître indûment la taille de stockage.

De plus, dans leur forme simple, la paramétrisation des graphes de groupes de mouvements souffre du même problème que les *fat graphs* [SO06], soit que l'enchaînement de deux mouvements doit transiter à travers une pose spécifique. En d'autres termes, la liberté supplémentaire permise par la combinaison d'animations est perdue lors de la transition d'un mouvement à un autre (figure 7.14(a)). Pour pallier à ce problème, Heck et Gleicher [HG07] proposent d'identifier, à l'aide d'une approximation par boîtes englobantes, une correspondance entre les pondérations utilisées pour le mouvement sortant et celles utilisées pour le mouvement entrant.

Dans notre cas, nous envisageons d'appliquer automatiquement la synthèse de transitions présentée à la section 7.5.2 aux mouvements obtenus par sur-échantillonnage (figure 7.14(c)). Nous pourrons ainsi générer un grand nombre de transitions entre deux mouvements particu-



(a) (b) (c)

FIG. 7.14 – Paramétrisation d'un groupe de mouvements par échantillonnage des combinaisons. (a) À l'intérieur d'un groupe, les mouvements peuvent être combinés, mais la transition d'un mouvement à un autre doit se faire à travers une pose spécifique, ici le groupe dégénéré J. (b) On peut cependant enrichir la banque de données en échantillonnant différentes combinaisons de mouvements [KG04b]. (c) Ensuite, la technique de la section 7.5.2 permet de synthétiser automatiquement de nouvelles transitions. L'enchaînement peut maintenant se faire à travers un grand nombre de nouvelles transitions, ou même utiliser des combinaisons de celles-ci.

liers, tel qu'illustré à la figure 7.14. Ces nouveaux mouvements, conservés explicitement à faible coût grâce à la compression, sont autant de points de transitions possibles. De plus, nous pouvons utiliser l'approximation et la technique de correction de Heck et Gleicher [HG07] pour obtenir un spectre continu de points de transition.

7.5.4 Synthèse interactive

On peut tracer de nombreux parallèles entre les graphes de groupes de mouvements et les arbres de mouvements (*move trees*) [Men00, MBC01]. Ces derniers sont des structures fréquemment utilisées dans l'industrie du jeu vidéo et permettent la synthèse en temps réel d'un mouvement répondant interactivement aux requêtes de l'utilisateur. Il est donc naturel de s'interroger sur la possibilité d'utiliser la structure de graphe proposée dans des applications semblables.

Lorsqu'on étudie l'algorithme de synthèse présenté à la section 7.3, on constate que celui-ci demande de connaître à l'avance l'enchaînement de mouvements souhaité. Il n'est pas difficile d'adapter cet algorithme à une situation où seul le successeur immédiat du mouvement en cours d'exécution est connu. Malgré cela, on constate qu'il se prête mal au contexte d'une application interactive où l'utilisateur peut décider à tout moment, en appuyant sur un bouton par exemple, du prochain mouvement à exécuter.

Pour contourner ce problème, on peut remarquer que la combinaison d'animations, illustrée

à la figure 7.5, n'a pas à s'exécuter sur toute la durée du mouvement. En effet, lorsque les mouvements sont assez similaires, une combinaison rapide n'introduira pas d'artéfacts apparents. C'est d'ailleurs l'approche préconisée par la plupart des techniques de graphes de mouvements, où la combinaison est effectuée uniquement sur les quelques poses précédant et suivant le point de transition [KGP02, GSKJ03, SO06]. Une approche possible pourrait donc être de prévoir une transition par défaut, par exemple vers un mouvement de repos (*idle*), et d'effectuer une combinaison rapide si ce comportement doit être modifié par une requête de l'usager.

Ceci nous amène à un second problème : le besoin de s'assurer que le graphe est assez riche pour permettre des transitions rapides vers les différents mouvements. En effet, si un comportement précis est attendu suite à une opération de l'usager, la nature interactive de l'application exige que ce mouvement s'exécute dans des délais assez courts. Dans la structure de graphes proposée par Gleicher *et al.* [GSKJ03], ceci est rendu possible en s'assurant de la présence d'un petit nombre de *hub nodes* : des noeuds fortement connectés d'où partent de courts mouvements. Cependant, compte tenu de la nature des graphes noeud-pose, ceci implique que les poses correspondant aux *hub nodes* réapparaîtront constamment lors de l'exécution de l'animation synthétisée.

Dans le cas des graphes de groupes de mouvements, une situation similaire peut être obtenue en s'assurant que la banque de données contienne un grand nombre de répétitions d'un ou de plusieurs mouvements de repos (*idle*). Par exemple, pour un jeu vidéo de boxe, on pourrait imaginer un mouvement de repos subtil autour de la pose neutre du boxeur. Ensuite, l'acteur peut exécuter des transitions de ce mouvement de repos vers tous les gestes qui peuvent être lancés interactivement. Dépendant des besoins, il est aussi possible d'inclure des enchaînements directs de ces gestes. Ceci pourrait permettre, par exemple, des combinaisons de coups de poing n'ayant pas à transiter par le mouvement de repos.

Le graphe résultant de cette situation contiendrait un noeud fortement connecté : le groupe de mouvements de repos. De celui-ci partiraient un grand nombre d'arcs correspondant aux gestes qui peuvent s'exécuter interactivement. Notons aussi que, lorsque combinée aux techniques des sections 7.5.2 et 7.5.3, l'animation synthétisée interactivement à partir de ce graphe n'aurait pas à transiter constamment par une même pose, au contraire de la technique de Gleicher *et al.* [GSKJ03]. De plus, le mouvement de repos pourrait être paramétré de façon à ce que les répétitions présentent de subtiles variations.

7.6 Conclusion

Dans ce chapitre, nous nous sommes attardés aux plus récents résultats obtenus suite à nos travaux de recherche. Ceux-ci comprennent l'organisation des sous-séquences redondantes, extraites grâce à l'algorithme présenté au chapitre 5, en un graphe de groupes de mouvements. Nous avons aussi montré comment ce graphe peut être utilisé pour synthétiser efficacement de nouvelles animations.

Finalement, nous avons discuté en détail de plusieurs directions de recherche combinant les graphes de groupes de mouvements et les techniques de compression du chapitre 6. Ces dernières observations permettent de constater l'étendue des applications possibles de la compression de banques de données d'animations.

Chapitre 8

Conclusion

I have every intention of writing a book – just as you advise – only I fear it will be many years before I have leisure enough to undertake it.

Mr Norrell

Que ce soit au cinéma ou dans les jeux vidéo, les animations de personnages humains sont aujourd'hui une facette essentielle de l'infographie. Pour les générer, on fait de plus en plus souvent appel à la capture de mouvements. Un des avantages de cette technique est qu'elle permet d'acquérir rapidement des animations adaptées à une situation précise. Cet avantage peut toutefois s'avérer un inconvénient important. En effet, l'utilisation fréquente de la capture de mouvements mène à une explosion de la quantité de données générées. Il devient donc important d'envisager différentes techniques permettant de réduire la taille de ces données. Ceci dans le but non seulement d'en faciliter le stockage, mais aussi de rendre possibles différentes applications qui peuvent bénéficier d'un accès efficace à une vaste banque de données.

Il existe plusieurs façons d'aborder le problème de la compression de données d'animation acquises par capture de mouvements. On peut tout d'abord envisager une compression sans pertes, qui permet de reproduire exactement les animations compressées, ou plutôt une compression avec pertes, qui tente d'exploiter les faiblesses de la perception humaine pour introduire des distorsions imperceptibles et ainsi accroître le taux de compression.

Un autre aspect à envisager lors du développement d'une technique de compression vient de la présence de deux types de cohérences dans les données d'animation. On sait d'une part que les degrés de liberté d'un squelette sont souvent corrélés. D'autre part, on remarque que les poses du squelette évoluent graduellement dans le temps sans présenter de rupture brusque. Dans une grande banque de données on peut même voir se répéter des sous-séquences entières.

Dans cette thèse, nous nous sommes penchés particulièrement sur les compressions avec pertes. De plus, nous avons choisi de ne pas nous attarder directement aux corrélations entre les degrés de liberté, préférant nous concentrer sur les façons d'exploiter les cohérences temporelles. Ces cohérences se présentent à différentes échelles : à petite échelle on remarque que les signaux d'animation sont généralement lisses et à grande échelle on remarque que certaines parties des animations sont fréquemment répétées. Nous nous sommes intéressés à ces deux types de cohérences.

Au chapitre 4, nous avons montré comment les cohérences à petite échelle pouvaient être exploitées dans le cadre d'une compression à base de transformées en ondelettes adaptée à la nature des animations squelettiques. Pour mettre au point cette compression, nous introduisons une technique d'optimisation explorant un espace de recherche raisonnable de façon à identifier les coefficients d'ondelettes à conserver pour minimiser la distorsion positionnelle. Cette distorsion, une approximation assez grossière des erreurs perceptibles introduites dans l'animation, ignore plusieurs facteurs affectant la fidélité visuelle de l'animation reconstruite. Ainsi, pour réduire les artéfacts liés au glissement des pieds sur le sol, nous proposons aussi une technique de correction basée sur la cinématique inverse.

Au chapitre 5 nous avons proposé une technique permettant d'identifier automatiquement les sous-séquences redondantes dans une banque de données. La clef de notre approche est l'exécution conjointe, dans un processus fortement couplé, de la segmentation et de l'agrégation de mouvements. Ceci nécessite un grand nombre d'opérations de recherche de sous-séquences. Pour accélérer ces recherches, nous utilisons une représentation simplifiée de la banque de données d'animations sous la forme d'une chaîne de caractères. Nous proposons aussi une façon efficace de déterminer si deux sous-chaînes correspondent à des sous-séquences semblables.

Ensuite, au chapitre 6, nous introduisons une technique de *PCA* quantifiée permettant d'utiliser les sous-séquences redondantes dans le cadre d'une technique de compression. Contrairement aux approches par *PCA* standard, l'utilisation de la *PCA* quantifiée permet le stockage simultané de la base vectorielle réduite et des données compressées. De plus, la segmentation utilisée, basée sur les sous-séquences redondantes, offre de nombreux avantages en comparaison de la segmentation statique précédemment proposée par Arikan [Ari06] pour la compression de banques de données.

Finalement, au chapitre 7, nous montrons comment l'identification des cohérences à grande échelle peut permettre de représenter une banque de données d'animations sous la forme d'un

graphe compact. Nous montrons ensuite comment ce graphe peut être utilisé, conjointement avec la combinaison d'animations (*blending*), pour synthétiser de nouvelles séquences.

8.1 Travaux futurs

Les travaux présentés dans cette thèse s'inscrivent dans un projet de recherche plus vaste qui se poursuivra au-delà du doctorat. En ce sens, nous explorons un certain nombre de directions qui permettraient de généraliser nos résultats ou encore de les exploiter dans des applications particulières. Les projets de recherche actuellement en cours et pour lesquels nous disposons de pistes précises sont présentés en détail dans le corps de la thèse : à la section 6.5 en ce qui a trait à l'utilisation des groupes de mouvements pour la compression, et à la section 7.5 en ce qui concerne la création de graphes et leur utilisation pour la synthèse d'animations.

Dans la suite de cette section nous brossons un rapide portrait de projets porteurs qui pourraient découler, directement ou indirectement, des contributions présentées dans cette thèse.

8.1.1 Métrique perceptuelle

Un problème orthogonal à cette thèse mais dont l'importance est indéniable pour l'avancement des techniques de compression avec pertes vient de l'absence d'une métrique efficace pour évaluer la distorsion perceptuelle des animations compressées. Une telle métrique est difficile à mettre au point car elle devrait dépendre de plusieurs facteurs extérieurs à l'animation. Le placement de la caméra, l'expertise de l'observateur, l'éclairage synthétique et la scène entourant l'animation ne sont que quelques-uns des éléments qui devraient être considérés. Dans cette thèse, nous avons proposé un certain nombre de métriques approximatives (voir section 2.3). En particulier, la distorsion positionnelle s'est avérée très utile pour la mise au point et l'évaluation des techniques de compression. Il n'en reste pas moins que cette métrique est grandement perfectible.

À ce sujet, nous avons d'ailleurs remarqué deux situations où la distorsion positionnelle se dégradait alors qu'une inspection visuelle permettait d'observer une amélioration de la qualité des animations. Tout d'abord, à la section 4.6, nous avons proposé une technique de correction des pieds par cinématique inverse qui permettait d'éliminer les artéfacts perceptibles liés au glissement des pieds sur le sol. Ensuite, à la section 6.4, nous avons remarqué que l'utilisation de la combinaison d'animations permettait d'éliminer les discontinuités perceptibles à la jonction

de deux sous-séquences. Dans ces deux cas, cependant, on remarquait une augmentation de la distorsion positionnelle.

Ainsi, nous croyons qu'il pourrait être possible de mettre au point une métrique plus efficace que la distorsion positionnelle pour prédire la qualité perçue d'une animation reconstruite. Cette métrique nous fournirait non seulement une meilleure façon d'évaluer les résultats obtenus, mais pourrait aussi être utilisée conjointement avec l'algorithme de sélection de coefficients proposé à la section 4.5. Elle pourrait de plus offrir une base plus solide pour la comparaison de différentes techniques.

8.1.2 Traitement de très grandes banques de données

La technique d'identification de redondances présentée au chapitre 5 s'avère très efficace pour traiter des banques de données de quelques minutes. Cependant, l'utilisation d'une technique d'agrégation demandant la présence en mémoire de toutes les données fait en sorte qu'il nous a été impossible de tester notre approche sur des banques de données de plusieurs heures. Le recours au sous-échantillonnage ou à une technique d'agrégation en-ligne (*online clustering*) pourrait nous permettre de contourner ce problème et ainsi d'étudier le comportement de notre approche sur un très grand nombre de données.

L'augmentation de la taille de la banque de données traitée pourrait avoir plusieurs répercussions importantes sur différents aspects de notre recherche. Par exemple, nous avons vu à la section 6.5.3 que l'utilisation de la *PCA* sur les poses ne permettait pas une réduction sensible de la taille de stockage. Dans de grandes banques de données, cependant, le nombre important de mouvements répétés pourrait invalider cette observation et la *PCA* pourrait s'avérer très efficace.

8.1.3 Groupes de mouvements séparables

Pour créer des groupes de mouvements (chapitre 5), nous nous intéressons uniquement aux gestes répétés qui impliquent tous les membres de l'acteur. Or, on sait qu'il existe des gestes se limitant à un sous-ensemble des membres. Par exemple, un personnage peut envoyer la main en restant sur place ou en marchant.

Nous croyons ainsi qu'il serait intéressant de mettre au point une technique d'identification de redondances séparables. Une telle technique serait en mesure d'identifier à la fois un mouvement répété impliquant tout l'acteur et un autre qui impliquerait un sous-ensemble de

ses membres. À ce sujet, nous pourrions envisager l'utilisation d'une hiérarchie prédéterminée de membres issue de connaissances *a priori* sur le squelette. Cependant, il pourrait aussi être possible de déterminer les séparations dynamiquement lorsqu'un nombre assez grand de sous-séquences redondantes sont observées sur un sous-ensemble des degrés de liberté.

Ikemoto et Forsyth [IF04] ont déjà montré le potentiel de la séparation des membres pour la synthèse de nouvelles animations. De plus, des techniques similaires sont déjà utilisées dans les jeux vidéo, où les différents membres d'un personnage animé sont parfois traités indépendamment. Nous pourrions ainsi étudier l'utilisation de groupes de mouvements séparables pour la synthèse de mouvements. Dans un tel cas, cependant, l'approche par graphe présentée au chapitre 7 ne pourrait pas s'appliquer directement étant donné que ces graphes ne permettent pas d'exprimer les contraintes régissant l'exécution simultanée de mouvements sur différents membres.

8.1.4 Création automatique de graphes paramétriques

À la fin du chapitre 7, nous avons discuté de différentes façons d'exploiter conjointement nos méthodes de compression et la technique de synthèse d'animations à partir de graphes de groupes de mouvements. Nous croyons que cette direction est très prometteuse et nous comptons nous y attarder dans le futur.

En particulier, nous souhaitons mettre au point une technique permettant d'enrichir la banque de données en synthétisant automatiquement des transitions entre les mouvements. Ceci permettrait de générer un graphe de groupes de mouvements allant au-delà des enchaînements réalisés par l'acteur. De plus, en combinant ceci à une technique de paramétrisation par sur-échantillonnage, nous pourrions obtenir des graphes paramétriques riches et flexibles. Il serait ensuite possible d'utiliser ces graphes pour la synthèse d'animations originales, aussi bien en présence d'un ensemble de buts à atteindre que dans le cadre d'une application interactive.

8.1.5 Création interactive de graphes

Présentement, la technique de création de graphes de groupes de mouvements est un processus qui s'effectue sur la banque de données après que tous les mouvements aient été capturés. Nous croyons cependant que certaines modifications pourraient permettre la création du graphe au fur et à mesure d'une séance de capture de mouvements. Un graphe généré interactivement

de la sorte pourrait faciliter la tâche d'un directeur en lui donnant une vue d'ensemble des mouvements et des transitions capturés par l'acteur.

Un tel outil permettrait, d'une part, de visualiser facilement les transitions manquantes et d'ajuster la séance de capture de mouvements en conséquence. D'autre part, le graphe ainsi généré rendrait possible la synthèse de nouveaux mouvements en temps réel, pendant la séance, permettant ainsi au directeur d'évaluer la richesse et la qualité de la banque de données capturée dans le contexte d'une utilisation particulière.

Bibliographie

[11172] ISO/IEC 11172-3. « Coding of moving pictures and associated audio for digital storage media at up to about 1,5 mbit/s, audio », 1993.

[13818] ISO/IEC 13818-7. « Generic coding of moving pictures and associated audio information, advanced audio coding (aac) », 2006.

[14491a] ISO/IEC 14491-1. « Coding of audio-visual objects, systems, amendment 1 », 1999.

[14491b] ISO/IEC 14491-2. « Coding of audio-visual objects, visual, amendment 1 », 1999.

[15444] ISO/IEC 15444-1. « Coding of still pictures, system », 2000.

[AC99] J. K. Aggarwal et Q. Cai. « Human motion analysis : A review ». *Computer Vision and Image Understanding*, volume 73, numéro 3, pages 428–440, 1999.

[ACCO05] J. Assa, Y. Caspi et D. Cohen-Or. « Action synopsis : Pose selection and illustration ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 24, numéro 3, pages 667–676, 2005.

[AF02] O. Arikan et D. A. Forsyth. « Interactive motion generation from examples ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 21, numéro 3, pages 483–490, 2002.

[AFO03] O. Arikan, D. A. Forsyth et J. F. O'Brien. « Motion synthesis from annotations ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 22, numéro 3, pages 402–408, 2003.

[AHM02] A. Ahmed, A. Hilton et F. Mokhtarian. « Adaptive compression of human animation data ». Dans *Eurographics '02 Short Presentations*, 2002.

[AKKH01] J.-H. Ahn, C.-S. Kim, C.-C. J. Kuo et Y.-S. Ho. « Motion compensated compression of 3D animation models ». *IEEE Electronics Letters*, volume 37, numéro 24, pages 1445–1446, 2001.

[ALP04] Y. Abe, C. K. Liu et Z. Popović. « Parameterization of dynamic character motion ». Dans *SCA '04 : Proc. Symp. Computer Animation*, pages 194–211, 2004.

[AM00] M. Alexa et W. Müller. « Representing animations by principal components ». *Computer Graphics Forum (Proc. Eurographics)*, volume 19, numéro 3, pages 411–418, 2000.

[Ari06] O. Arikan. « Compression of motion capture databases ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 25, numéro 3, pages 890–897, 2006.

[BBN00] G. Bachman, E. Beckenstein et L. Narici. *Fourier and wavelet analysis*. Springer, 2000.

[Bea07a] P. Beaudoin. « Compression de données d'animation acquises par capture de mouvements, Séquence vidéo 1 : Compression en ondelettes ». http ://www.iro.umontreal.ca/labs/infographie/theses/beaudoin-phd/, 2007.

[Bea07b] P. Beaudoin. « Compression de données d'animation acquises par capture de mouvements, Séquence vidéo 2 : Détection de sous-séquences redondantes, graphes et synthése ». http ://www.iro.umontreal.ca/labs/infographie/theses/beaudoin-phd/, 2007.

[Bea07c] P. Beaudoin. « Compression de données d'animation acquises par capture de mouvements, Séquence vidéo 3 : Compression de sous-séquences redondantes ». http ://www.iro.umontreal.ca/labs/infographie/theses/beaudoin-phd/, 2007.

[BH00] M. Brand et A. Hertzmann. « Style machines ». Dans *Proc. SIGGRAPH '00*, pages 183–192, 2000.

[Bin00] R. N. Bindiganavale. *Building parameterized action representations from observagion*. Thèse de doctorat, University of Pennsylvania, 2000.

[Bis95] C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[Bow00] R. Bowden. « Learning statistical models of human motion ». Dans *Proc. Intl. Workshop on Human Modeling, Analysis and Synthesis*, 2000.

[BP04] P. Beaudoin et P. Poulin. « Compressed multisampling for efficient hardware edge antialiasing ». Dans *GI '04 : Proc. Graphics Interface*, pages 169–176, mai 2004.

[BPv07a] P. Beaudoin, P. Poulin et M. van de Panne. « Adapting wavelet compression techniques to skeletal motion capture data ». Dans *SCA '07 Research posters*, 2007.

[BPv07b] P. Beaudoin, P. Poulin et M. van de Panne. « Adapting wavelet compression to human motion capture clips ». Dans *GI '07 : Proc. Graphics Interface*, pages 313–318, 2007.

[BSC05] S. Basu, S. Shanbhag et S. Chandran. « Search and transitioning for motion captured sequences ». Dans *VRST '05 : Proc. Symp. Virtual Reality Software and Technology*, pages 220–223, 2005.

[BSJ04] M. Bourges-Sévenier et E. S. Jang. « An introduction to the MPEG-4 animation framework extension (afx) ». *IEEE Trans. on Circuits and Systems for Video Technology*, volume 14, numéro 7, pages 928–936, 2004.

[BSM+03] H. M. Briceño, P. V. Sander, L. McMillan, S. Gortler et H. Hoppe. « Geometry videos : A new representation for 3D animations ». Dans *SCA '03 : Proc. Symp. Computer Animation*, pages 136–146, 2003.

[BSP+04] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins et N. S. Pollard. « Segmenting Motion Capture Data into Distinct Behaviors ». Dans *GI '04 : Proc. Graphics Interface*, pages 185–194, 2004.

[Bur98] C. J. C. Burges. « A tutorial on support vector machines for pattern recognition ». *Data Mining and Knowledge Discovery*, volume 2, numéro 2, pages 121–167, 1998.

[BvP07a] P. Beaudoin, M. van de Panne et P. Poulin. « Automated extraction of compact motion graphs ». Dans *SCA '07 Research posters*, 2007.

[BvP07b] P. Beaudoin, M. van de Panne et P. Poulin. « Automatic construction of compact motion graphs ». Rapport technique 1296, Département d'informatique et de recherche opérationnelle, Université de Montréal, mai 2007.

[BW95] A. Bruderlin et L. Williams. « Motion signal processing ». Dans *Proc. SIGGRAPH '95*, pages 97–104, 1995.

[CBL05] S. Chattopadhyay, S. M. Bhandarkar et K. Li. « BAP sparsing : A novel approach to MPEG-4 body animation parameter compression ». Dans *Proc. Systems Communications*, pages 104–109, 2005.

[CBL07] S. Chattopadhyay, S. M. Bhandarkar et K. Li. « Human motion capture data compression by model-based indexing : A power aware approach ». *IEEE Trans. on Visualization and Computer Graphics*, volume 13, numéro 1, pages 5–14, 2007.

[CDF92] A. Cohen, I. Daubechies et J.-C. Feauveau. « Biorthogonal bases of compactly supported wavelets ». *Commun. on Pure and Applied Mathematics*, volume 45, numéro 5, pages 485–560, 1992.

[CH05] J. Chai et J. K. Hodgins. « Performance animation from low-dimensional control signals ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 24, numéro 3, pages 686–696, 2005.

[Chu92a] C. K. Chui. *An introduction to wavelets*. Academic Press Professional, 1992.

[Chu92b] C. K. Chui, éditeur. *Wavelets : A tutorial in theory and applications*. Academic Press Professional, 1992.

[CLS03] M. G. Choi, J. Lee et S. Y. Shin. « Planning biped locomotion using motion capture data and probabilistic roadmaps ». *ACM Trans. Graphics*, volume 22, numéro 2, pages 182–203, 2003.

[CMU] Carnegie Mellon Universtity. « CMU graphics lab motion capture database ». http ://mocap.cs.cmu.edu/, janvier 2007.

[Coh92] M. F. Cohen. « Interactive spacetime control for animation ». Dans *Proc. SIGGRAPH '92*, pages 293–302, 1992.

[CPO00] T. K. Capin, E. Petajan et J. Ostermann. « Very low bitrate coding of virtual human animation in MPEG-4 ». Dans *ICME '00 : Proc. Intl. Conf. on Multimedia and Expo*, volume 2, pages 1107–1110, 2000.

[CT99] T. K. Capin et D. Thalmann. « Controlling and efficient coding of MPEG-4 compliant avatars ». Dans *Proc. Intl. Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging*, 1999.

[Dal93] S. Daly. « The visible differences predictor : an algorithm for the assessment of image fidelity ». *Digital images and human vision*, pages 179–206, 1993.

[Dau92] I. Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, 1992.

[DP86] R. Demori et D. Probst. *Handbook of pattern recognition and image processing*. Academic Press, 1986.

[Dro02] A. Drozdek. *Elements of data compression*. Brooks/Cole Thomson Learning, 2002.

[Dun04] Q. Dunki. « Ai development for full spectrum warrior ». Montreal Game Summit, novembre 2004.

[EON04] H. Etou, Y. Okada et K. Niijima. « Feature preserving motion compression based on hierarchical curve simplification ». Dans *ICME '04 : Proc. Intl. Conf. on Multimedia and Expo*, volume 2, pages 1435–1438, 2004.

[FBCM04] C. Fowlkes, S. Belongie, F. Chung et J. Malik. « Spectral grouping using the nystrom method ». *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 26, numéro 2, pages 214–225, 2004.

[Fěd03] M. Fědor. « Application of inverse kinematics for skeleton manipulation in real-time ». Dans *SCCG '03 : Proc. Spring Conference on Computer Graphics*, pages 203–212, 2003.

[FF05] K. Forbes et E. Fiume. « An efficient search algorithm for motion data using weighted pca ». Dans *SCA '05 : Proc. Symp. Computer Animation*, pages 67–76, 2005.

[FMJ02] A. Fod, M.J. Mataric et O. C. Jenkins. « Automated derivation of primitives for movement classification ». *Autonomous Robot*, volume 12, numéro 1, pages 39–54, 2002.

[GBT04] P. Glardon, R. Boulic et D. Thalmann. « PCA-based walking engine using motion capture data ». Dans *CGI '04 : Proc. Computer Graphics International*, pages 292–298, 2004.

[GGH02] X. Gu, S. J. Gortler et H. Hoppe. « Geometry images ». Dans *Proc. SIGGRAPH '02*, pages 355–361, 2002.

[GK04] I. Guskov et A. Khodakovsky. « Wavelet compression of parametrically coherent mesh sequences ». Dans *SCA '04 : Proc. Symp. Computer Animation*, pages 183–192, 2004.

[GMHP04] K. Grochow, S. L. Martin, A. Hertzmann et Z. Popović. « Style-based inverse kinematics ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 23, numéro 3, pages 522–531, 2004.

[Gra98] F. S. Grassia. « Practical parameterization of rotations using the exponential map ». *Journal of Graphics Tools*, volume 3, numéro 3, pages 29–48, 1998.

[GSKJ03] M. Gleicher, H.J. Shin, L. Kovar et A. Jepsen. « Snap-together motion : Assembling runtime animations ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 22, numéro 3, pages 702–702, 2003.

[HB00] H. H. Hecht et M. Bertamini. « Understanding projectile acceleration ». *Journal of Experimental Psychology : Human Perception and Performance*, volume 26, numéro 2, pages 730–746, 2000.

[Het04] M.L. Hetland. « A survey of recent methods for efficient retrieval of similar time sequences ». Dans M. Last, A. Kandel et H. Bunke, éditeurs. *Data Mining in Time Series Databases*. World Scientific, 2004.

[HG07] R. Heck et M. Gleicher. « Parametric motion graphs ». Dans *I3D '07 : Proc. Symp. Interactive 3D Graphics and Games*, pages 129–136, 2007.

[HOT98] J. K. Hodgins, J. F. O'Brien et J. Tumblin. « Perception of human motion with different geometric models ». *IEEE Trans. on Visualization and Computer Graphics*, volume 4, numéro 4, pages 307–316, 1998.

[HRv03] J. Harrison, R. A. Rensink et M. van de Panne. « Detecting changes of velocity of smoothly moving objects ». *Journal of Vision*, volume 3, numéro 9, page 401, 2003.

[HRv04] J. Harrison, R. A. Rensink et M. van de Panne. « Obscuring length changes during animated motion ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 23, numéro 3, pages 569–573, 2004.

[Huf51] D. A. Huffman. « A method for the construction of minimum redundancy codes ». Dans *Proc. of the Institute of Radio Engineers*, volume 40, pages 1098–1101, 1951.

[IAF06] L. Ikemoto, O. Arikan et D. A. Forsyth. « Knowing when to put your foot down ». Dans *I3D '06 : Proc. Symp. Interactive 3D Graphics and Games*, pages 49–53, 2006.

[IAF07] L. Ikemoto, O. Arikan et D. A. Forsyth. « Quick transitions with cached multi-way blends ». Dans *I3D '07 : Proc. Symp. Interactive 3D Graphics and Games*, pages 145–151, 2007.

[IF04] L. Ikemoto et D. A. Forsyth. « Enriching a motion collection by transplanting limbs ». Dans *SCA '04 : Proc. Symp. Computer Animation*, pages 99–108, 2004.

[IR03] L. Ibarria et J. Rossignac. « Dynapack : Space-time compression of the 3D animations of triangle meshes with fixed connectivity ». Dans *SCA '03 : Proc. Symp. Computer Animation*, pages 126–135, 2003.

[JM03] O. C. Jenkins et M. J. Mataric. « Automated derivation of behavior vocabularies for autonomous humanoid motion ». Dans *AAMAS '03 : Proc. Intl. Conf. Autonomous agents and multiagent systems*, pages 225–232, 2003.

[Joh73] G. Johansson. « Visual perception of biological motion and a model for its analysis ». *Perception and Psychophysics*, volume 14, numéro 2, 1973.

[Jol86] I. Joliffe. *Principal component analysis*. Springer-Verlag, 1986.

[JT05] D. L. James et C. D. Twigg. « Skinning mesh animations ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 24, numéro 3, pages 399–407, 2005.

[KG03] L. Kovar et M. Gleicher. « Flexible automatic motion blending with registration curves ». Dans *SCA '03 : Proc. Symp. Computer Animation*, pages 214–224, 2003.

[KG04a] Z. Karni et C. Gotsman. « Compression of soft-body animation sequences ». *Computers & Graphics*, volume 28, numéro 1, pages 25–34, 2004.

[KG04b] L. Kovar et M. Gleicher. « Automated extraction and parameterization of motions in large data sets ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 23, numéro 3, pages 559–568, 2004.

[KGP02] L. Kovar, M. Gleicher et F. Pighin. « Motion graphs ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 21, numéro 3, pages 473–482, 2002.

[KLS00] T.-H. Kim, J. Lee et S. Y. Shin. « Robust motion watermarking based on multiresolution analysis ». *Computer Graphics Forum (Proc. Eurographics)*, volume 19, numéro 3, pages 189–198, 2000.

[KM04] K. Kondo et K. Matsuda. « Keyframes extraction method for motion capture data ». *Journal for Geometry and Graphics*, volume 8, numéro 1, pages 81–90, 2004.

[Kou95] W. Kou. *Digital image compression : Algorithms and standards*. Springer, 1995.

[KP97] B.-J. Kim et W. A. Pearlman. « An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (spiht) ». Dans *DCC '97 : Proc. Data Compression Conference*, page 251, 1997.

[KPS03] T.-H. Kim, S. I. Park et S. Y. Shin. « Rhythmic-motion synthesis based on motion-beat analysis ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 22, numéro 3, pages 392–401, 2003.

[KPZ+04] E. Keogh, T. Palpanas, V.B. Zordan, D. Gunopulos et M. Cardle. « Indexing large human-motion databases ». Dans *VLDB '04 : Proc. Very Large Data Bases Conf.*, pages 780–791, 2004.

[KS05] T. Kwon et S. Y. Shin. « Motion modeling for on-line locomotion synthesis ». Dans *SCA '05 : Proc. Symp. Computer Animation*, pages 29–38, 2005.

[KSG02] L. Kovar, J. Schreiner et M. Gleicher. « Footskate cleanup for motion capture editing ». Dans *SCA '02 : Proc. Symp. Computer Animation*, pages 97–104, 2002.

[KTP04] K. Kahol, P. Tripathi et S. Panchanathan. « Automated gesture segmentation from dance sequences ». Dans *Proc. Intl. Conf. on Automatic Face and Gesture Recognition*, pages 883–888, 2004.

[Lav04] S. Lavier. « Les besoins en compression d'animations pour les jeux vidéo chez ubisoft ». Communication privée, octobre 2004.

[Law04] N. D. Lawrence. « Gaussian process latent variable models for visualisation of high dimensional data ». *Advances in Neural Information Processing Systems*, numéro 16, pages 329–336, 2004.

[LCF00] J. P. Lewis, M. Cordner et N. Fong. « Pose space deformation : A unified approach to shape interpolation and skeleton-driven deformation ». Dans *Proc. SIGGRAPH '00*, pages 165–172, 2000.

[LCF05] Y.-C. Lai, S. Chenney et S. Fan. « Group motion graphs ». Dans *SCA '05 : Proc. Symp. Computer Animation*, pages 281–290, 2005.

[LCR+02] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins et N. S. Pollard. « Interactive control of avatars animated with human motion data ». Dans *Proc. SIGGRAPH '02*, pages 491–500, New York, NY, USA, 2002. ACM Press.

[Len99] J. E. Lengyel. « Compression of time-dependent geometry ». Dans *I3D '99 : Proc. Symp. Interactive 3D Graphics*, pages 89–95, 1999.

[LGC94] Z. Liu, S.J. Gortler et M. F. Cohen. « Hierarchical spacetime control ». Dans *Proc. SIGGRAPH '94*, pages 35–42, 1994.

[Lit91] P. C. Litwinowicz. « Inkwell : A 2.5-D animation system ». Dans *Proc. SIGGRAPH '91*, pages 113–122, 1991.

[LL06a] C.-H. Lee et J. Lasenby. « 3D human motion compression using wavelet decomposition ». Dans *SIGGRAPH '06 Research posters*, 2006.

[LL06b] J. Lee et K. H. Lee. « Precomputing avatar behavior from human motion data ». *Graphical Models*, volume 68, numéro 2, pages 158–174, 2006.

[LM06] G. Liu et L. McMillan. « Segment-based human motion compression ». Dans *SCA '06 : Proc. Symp. on Computer Animation*, pages 127–135, 2006.

[LOT04] S. Li, M. Okuda et S. Takahashi. « Hierarchical human motion compression with constraints on frames ». Dans *MWSCAS '04 : Proc. Midwest Symp. on Circuits and Systems*, volume 1, pages 253–256, 2004.

[LOT05] S. Li, M. Okuda et S. Takahashi. « Kinematics based motion compression for human figure animation ». Dans *ICASSP '05 : Proc. Intl. Conf. Acoustics, Speech, and Signal Processing*, volume 2, pages 1077–1080, 2005.

[LS99] J. Lee et S. Y. Shin. « A hierarchical approach to interactive motion editing for human-like figures ». Dans *Proc. SIGGRAPH '99*, pages 39–48, 1999.

[LS00] J. Lee et S. Y. Shin. « Multiresolution motion analysis with applications ». Dans *Proc. Intl. Workshop on Human Modeling and Animation*, pages 131–143, 2000.

[LS01] J. Lee et S. Y. Shin. « A coordinate-invariant approach to multiresolution motion analysis ». *Graphical Models*, volume 63, numéro 2, pages 87–105, 2001.

[LS02] J. Lee et S. Y. Shin. « General construction of time-domain filters for orientation data ». *IEEE Trans. on Visualization and Computer Graphics*, volume 8, numéro 2, pages 119–128, 2002.

[LT01] I. S. Lim et D. Thalmann. « Key-posture extraction out of human motion data by curve simplification ». Dans *Proc. Intl. Conf. Engineering in Medicine and Biology Society*, volume 2, pages 1167–1169, 2001.

[Lub97] J. Lubin. « A human vision system model for objective picture quality measurements ». Dans *Proc. Intl. Broadcasting Convention*, pages 498–503, 1997.

[LWS97] S. Lee, G. Wolberg et S. Y. Shin. « Scattered data interpolation with multilevel B-splines ». *IEEE Trans. on Visualization and Computer Graphics*, volume 3, numéro 3, pages 228–244, 1997.

[LWS02] Y. Li, T. Wang et H.-Y. Shum. « Motion texture : a two-level statistical model for character motion synthesis ». Dans *Proc. SIGGRAPH '02*, pages 465–472, 2002.

[LZWM05] G. Liu, J. Zhang, W. Wang et L. McMillan. « A system for analyzing and indexing human-motion databases ». Dans *SIGMOD '05 : Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pages 924–926, 2005.

[LZWM06] G. Liu, J. Zhang, W. Wang et L. McMillan. « Human motion estimation from a reduced marker set ». Dans *I3D '06 : Proc. Symp. Interactive 3D Graphics and Games*, pages 35–42, 2006.

[LZWP03] F. Liu, Y. Zhuang, F. Wu et Y. Pan. « 3D motion retrieval with motion index tree ». *Computer Vision and Image Understanding*, volume 92, numéro 2-3, pages 265–284, 2003.

[Mah36] P. C. Mahalanobis. « On the generalized distance in statistics ». Dans *Proc. of the National Institute of Science of India*, volume 12, pages 49–55, 1936.

[Mal98] S. Mallat. *A wavelet tour of signal processing*. Academic Press, 1998.

[MBC01] M. Mizuguchi, J. Buchanan et T. Calvert. « Data driven motion transitions for interactive games ». Dans *Eurographics '01 Short Presentations*, 2001.

[McG68] V. C. McGee. « Multidimensional scaling of n sets of similarity measures : A nonmetric individual differences approach ». *Multivariate Behaviour Research*, volume 3, pages 233–248, 1968.

[Men00] A. Menache. *Understanding motion capture for computer animation and video games*. Academic Press, 2000.

[MK05] T. Mukai et S. Kuriyama. « Geostatistical motion interpolation ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 24, numéro 3, pages 1062–1070, 2005.

[MKMA04] S. Ménardais, R. Kulpa, F. Multon et B. Arnaldi. « Synchronization for dynamic blending of motions ». Dans *SCA '04 : Proc. Symp. Computer Animation*, pages 325–335, 2004.

[MM05] M. Meredith et S. Maddock. « Adapting motion capture data using weighted real-time inverse kinematics ». *Computers in Entertainment*, volume 3, numéro 1, pages 5–19, 2005.

[MR06] M. Müller et T. Röder. « Motion templates for automatic classification and retrieval of motion capture data ». Dans *SCA '06 : Proc. Symp. Computer Animation*, pages 137–146, 2006.

[MRC05] M. Müller, T. Röder et M. Clausen. « Efficient content-based retrieval of motion capture data ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 24, numéro 3, pages 677–685, 2005.

[OD01] C. O'Sullivan et J. Dingliana. « Collisions and perception ». *ACM Trans. Graphics*, volume 20, numéro 3, pages 151–168, 2001.

[ODGK03] C. O'Sullivan, J. Dingliana, T. Giang et M. K. Kaiser. « Evaluating the visual fidelity of physically based animations ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 22, numéro 3, pages 527–536, 2003.

[OHJ00] M. Oesker, H. Hecht et B. Jung. « Psychological evidence for unconscious processing of detail in real-time animation of multiple characters ». *Journal of Visualization and Computer Animation*, volume 11, numéro 2, pages 105–112, 2000.

[ORC99] C. O'Sullivan, R. Radach et S. Collins. « A model of collision perception for real-time animation ». Dans *CAS '99 : Proc. of the Eurographics Workshop in Computer Animation and Simulation*, pages 67–76, 1999.

[PB00] K. Pullen et C. Bregler. « Animating by multi-level sampling ». Dans *CA '00 : Proc. Computer Animation*, pages 36–42, 2000.

[PB02] K. Pullen et C. Bregler. « Motion capture assisted animation : Texturing and synthesis ». Dans *Proc. SIGGRAPH '02*, pages 501–508, 2002.

[Per95] K. Perlin. « Real time responsive animation with personality ». *IEEE Trans. on Visualization and Computer Graphics*, volume 1, numéro 1, pages 5–15, 1995.

[PHM03] F. Pollick, J. G. Hale et P. McAleer. « Visual perception of humanoid movement ». Dans *Proc. Intl. Workshop on Epigenetic Robotics*, pages 107–114, 2003.

[PI97] L. Prasad et S. S. Iyengar. *Wavelet analysis with applications to image processing*. CRC Press, 1997.

[PRM00] V. Pavlovic, J. M. Rehg et J. MacCormick. « Learning switching linear models of human motion ». Dans *NIPS '00 : Proc. Neural Information Processing Systems*, pages 981–987, 2000.

[PSKS04] S. I. Park, H.J. Shin, T.-H. Kim et S. Y. Shin. « On-line motion blending for real-time locomotion generation ». *Computer Animation and Virtual Worlds*, volume 15, numéro 3-4, pages 125–138, 2004.

[PSS02] S. I. Park, H.J. Shin et S. Y. Shin. « On-line locomotion generation based on motion blending ». Dans *SCA '02 : Proc. Symp. Computer Animation*, pages 105–111, 2002.

[PSS04] S. I. Park, H.J. Shin et S. Y. Shin. « Online motion blending for real-time locomotion generation ». *Computer Animation and Virtual Worlds*, volume 15, numéro 3-4, pages 125–138, 2004.

[Rab89] L. R. Rabiner. « A tutorial on hidden markov models and selected applications in speech recognition ». *Proceedings of the IEEE*, volume 77, numéro 2, pages 257–286, 1989.

[RCB98] C. Rose, M. F. Cohen et B. Bodenheimer. « Verbs and adverbs : Multidimensional motion interpolation ». *IEEE Computer Graphics and Applications*, volume 18, numéro 5, pages 32–40, 1998.

[RGBC96] C. Rose, B. Guenter, B. Bodenheimer et M. F. Cohen. « Efficient generation of motion transitions using spacetime constraints ». Dans *Proc. SIGGRAPH '96*, pages 147–154, 1996.

[Ris76] J. J. Rissanen. « Generalized kraft inequality and arithmetic coding ». *IBM Journal of Research and Development*, volume 20, pages 198–203, 1976.

[RL79] J. J. Rissanen et G. G. Langdon. « Arithmetic coding ». *IBM Journal of Research and Development*, volume 23, numéro 2, pages 149–162, 1979.

[Ros99] J. Rossignac. « Edgebreaker : Connectivity compression for triangle meshes ». *IEEE Trans. on Visualization and Computer Graphics*, volume 5, numéro 1, pages 47–61, 1999.

[RP03] P. S. A. Reitsma et N. S. Pollard. « Perceptual metrics for character animation : sensitivity to errors in ballistic motion ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 22, numéro 3, pages 537–542, 2003.

[RPE+05] L. Ren, A. Patrick, A. A. Efros, J. K. Hodgins et J. M. Rehg. « A data-driven approach to quantifying natural human motion ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 24, numéro 3, pages 1090–1097, 2005.

[RSC01] C. F. Rose, P.-P. Sloan et M. F. Cohen. « Artist-directed inverse-kinematics using radial basis function interpolation ». *Computer Graphics Forum (Proc. Eurographics)*, volume 20, numéro 3, pages 239–250, 2001.

[RWO98] S. M. Redl, M. K. Weber et M. W. Oliphant. *Gsm and personal communications handbook*. Artech House, 1998.

[Sal02] D. Salomon. *A guide to data compression methods*. Springer, 2002.

[Sal04] D. Salomon. *Data compression : The complete reference*. Springer, 3ième édition, 2004.

[Say96] K. Sayood. *Introduction to data compression*. Morgan Kaufmann, 1996.

[SB05a] A. E. Seward et B. Bodenheimer. « Using nonlinear dimensionality reduction in 3D figure animation ». Dans *ACM-SE '05 : Proc. annual Southeast regional Conf.*, pages 388–392, 2005.

[SB05b] C. K. F. So et G. Baciu. « Entropy-based motion extraction for motion capture animation ». *Computer Animation and Virtual Worlds*, volume 16, numéro 3-4, pages 225–235, 2005.

[SB06] C. K. F. So et G. Baciu. « Hypercube sweeping algorithm for subsequence motion matching in large motion databases ». Dans *VRCIA '06 : Proc. ACM Intl. Conf. Virtual Reality Continuum and its Applications*, pages 221–228, 2006.

[SDS96] E. J. Stollnitz, T. D. DeRose et D. H. Salesin. *Wavelets for computer graphics : Theory and applications*. Morgan Kaufmann, 1996.

[SH05] A. Safonova et J. K. Hodgins. « Analyzing the physical correctness of interpolated human motion ». Dans *SCA '05 : Proc. Symp. Computer Animation*, pages 171–180, 2005.

[SH07] A. Safonova et J. K. Hodgins. « Construction and optimal search of interpolated motion graphs ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 26, numéro 3, page 106, 2007.

[She92] M. J. Shensa. « The discrete wavelet transform : Wedding the À Trous and Mallat algorithms ». *IEEE Trans. on Signal Processing*, volume 40, numéro 10, pages 2464–2482, 1992.

[SHP04] A. Safonova, J. K. Hodgins et N. S. Pollard. « Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces ». *ACM Trans. Graphics (Proc. SIGGRAPH)*, volume 23, numéro 3, pages 514–521, 2004.

[SM00] J. Shi et J. Malik. « Normalized cuts and image segmentation ». *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 22, numéro 8, pages 228–244, 2000.

[SMM05] M. Srinivasan, R. A. Metoyer et E. N. Mortensen. « Controllable real-time locomotion using mobility maps ». Dans *GI '05 : Proc. Graphics Interface*, pages 51–59, 2005.

[SO06] H.J. Shin et H. S. Oh. « Fat graphs : Constructing an interactive character with continuous controls ». Dans *SCA '06 : Proc. Symp. Computer Animation*, pages 291–298, 2006.

[SS95] W. Sweldens et P. Schröder. « Building your own wavelets at home ». Rapport technique 1995.5, Mathematics Department, University of Southern California, 1995.

[SSK05] M. Sattler, R. Sarlette et R. Klein. « Simple and efficient compression of animation sequences ». Dans *SCA '05 : Proc. Symp. on Computer Animation*, pages 209–217, 2005.

[Swe96] W. Sweldens. « The lifting scheme : A custom-design construction of biorthogonal wavelets ». *Journal of Applied and Computational Harmonic Analysis*, volume 3, numéro 2, pages 186–200, 1996.

[Swe98] W. Sweldens. « The lifting scheme : A construction of second generation wavelets ». *SIAM Journal on Mathematical Analysis*, volume 29, numéro 2, pages 511–546, 1998.

[Sym03] P. D. Symes. *Digital video compression.* McGraw-Hill, 2003.

[TdL00] J. B. Tenenbaum, V. de Silva et J. C. Langford. « A global geometric framework for nonlinear dimensionality reduction ». *Science*, volume 290, numéro 5500, pages 2319–2323, 2000.

[TG98] C. Touma et C. Gotsman. « Triangle mesh compression ». Dans *GI '98 : Proc. Graphics Interface '98*, pages 26–34, 1998.

[TH94] P. Teo et D. Heeger. « Perceptual image distortion ». Dans *ICIP '94 : Proc. Intl. Conf. on Image Processing*, volume 2, pages 982–986, 1994.

[TH00] L. M. Tanco et A. Hilton. « Realistic synthesis of novel human movements from a database of motion capture examples ». Dans *HUMO '00 : Proc. Workshop on Human Motion*, pages 137–142, 2000.

[UAT95] M. Unuma, K. Anjyo et R. Takeuchi. « Fourier principles for emotion-based human figure animation ». Dans *Proc. SIGGRAPH '95*, pages 91–96, 1995.

[Uyt99] G. Uytterhoeven. *Wavelets : Software and applications*. Thèse de doctorat, Université Catholique de Louvain, 1999.

[van04] C. J. van den Branden Lambrecht. « Color moving pictures quality metric ». Dans *ICIP '96 : Proc. Intl. Conf. on Image Processing*, volume 1, pages 885–888, 2004.

[Wal91] G. K. Wallace. « The JPEG still picture compression standard ». *Commun. ACM*, volume 34, numéro 4, pages 30–44, 1991.

[Wal07] M. Walker. « Présentation pléniaire d'ouverture ». Graphics Interface, mai 2007.

[WB99] A. D. Wilson et A. F. Bobick. « Parametric hidden markov models for gesture recognition ». *Trans. Pattern Analysis and Machine Intelligence*, volume 21, numéro 9, pages 884–900, 1999.

[WB03] J. Wang et B. Bodenheimer. « An evaluation of a cost metric for selecting transitions between motion segments ». Dans *SCA '03 : Proc. Symp. on Computer Animation*, pages 232–238, San Diego, CA, 2003.

[WB04] J. Wang et B. Bodenheimer. « Computing the duration of motion transitions : An empirical approach ». Dans *SCA '04 : Proc. Symp. on Computer Animation*, pages 335–344, 2004.

[Wel93] C. Welman. « Inverse kinematics and geometric constraints for articulated figure manipulation ». Mémoire de maîtrise, Simon Fraser University, 1993.

[WFM01] B. Watson, A. Friedman et A. McGaffey. « Measuring and predicting visual fidelity ». Dans *Proc. SIGGRAPH '01*, pages 213–220, 2001.

[WH97] D. J. Wiley et J. K. Hahn. « Interpolation synthesis of articulated figure motion ». *IEEE Computer Graphics and Applications*, volume 17, numéro 6, pages 39–45, 1997.

[WK88] A. Witkin et M. Kass. « Spacetime constraints ». Dans *Proc. SIGGRAPH '88*, pages 159–168, 1988.

[WP95] A. Witkin et Z. Popović. « Motion warping ». Dans *Proc. SIGGRAPH '95*, pages 105–108, 1995.

[WS98] A. Watson et M. A. Sasse. « Measuring perceived quality of speech and video in multimedia conferencing applications ». Dans *MULTIMEDIA '98 : Proc. Intl. Conf. on Multimedia*, pages 55–60, 1998.

[YKL02] J.-H. Yang, C.-S. Kim et S.-U. Lee. « Compression of 3D triangle mesh sequences based on vertex-wise motion vector prediction ». *IEEE Trans. on Circuits and Systems for Video Technology*, volume 12, numéro 12, 2002.

[YKL04] J.-H. Yang, C.-S. Kim et S.-U. Lee. « Hybrid coding for animated polygonal meshes : Combining delta and octree ». Dans *ICIP '04 : Proc. Intl. Conf. on Image Processing*, 2004.

[Zha01] L. Zhao. *Synthesis and acquisition of laban movement analysis qualitative parameters for communicative gestures*. Thèse de doctorat, University of Pennsylvania, 2001.

[ZL77] J. Ziv et A. Lempel. « A universal algorithm for data compression ». *IEEE Transactions on Information Theory*, volume 23, numéro 3, pages 337–343, 1977.

[ZL78] J. Ziv et A. Lempel. « Compression of individual sequences via variable-rate coding ». *IEEE Transactions on Information Theory*, volume 24, numéro 5, pages 530–536, 1978.

[ZO04] J. Zhang et C. B. Owen. « Octree-based animated geometry compression ». Dans *DCC '04 : Proc. Data Compression Conf.*, 2004.

[ZRH98] Y. Zhuang, Y. Rui et T. S. Huang. « Adaptive key frame extraction using unsupervised clustering ». Dans *ICIP '04 : Proc. Intl. Conf. on Image Processing*, pages 866–870, 1998.