Techniques d'animation pour gérer les interactions entre un combattant virtuel et un sujet réel

Nicolas Pronost*
Bunraku project, IRISA, Univ. Rennes1

Franck Multon†
Bunraku project/Univ. Rennes2, IRISA

Qilei Li‡
State Key Lab. CAD&CG, Zhejiang Univ.

Weidong Geng§
State Key Lab. CAD&CG, Zhejiang Univ.

Richard Kulpa¶
M2S, Univ. Rennes 2

Georges Dumont‖
Bunraku project IRISA, ENS Cachan

ABSTRACT

Cet article propose une méthode pour animer des humains virtuels qui peuvent interagir efficacement avec des sujets réels en réalité virtuelle. Si les ordres du sujet peuvent être modélisés par des cibles à atteindre et des commandes, le système cherche le comportement le plus adapté dans une base de données. Afin d'éviter le recours à de volumineuses bases de données pour gérer tous les cas possibles, nous proposons d'associer à cette recherche un module d'adaptation de mouvements. En effet, même si le mouvement sélectionné ne correspond pas parfaitement à la situation, il peut être adapté dans le but de satisfaire précisément les contraintes données par l'utilisateur. Cette méthode est illustrée ici avec l'exemple du combattant virtuel de kung-fu. Deux personnes sont impliquées dans cet exemple : l'utilisateur et le superviseur. Le premier se déplace librement dans le monde réel tandis que la position de sa tête est obtenue en temps réel avec des marqueurs réfléchissants. Le combattant virtuel suit les déplacements de l'utilisateur dans le but de rester à une distance compatible avec un coup de pied ou de poing. A n'importe quel moment, le superviseur peut demander au combattant de frapper la tête de l'utilisateur avec ses pieds ou ses poings. Le système recherche automatiquement le mouvement le plus adéquate dans une petite base de données (16 mouvements, contrairement aux approche de type graphe de mouvements qui nécessitent plusieurs centaines de clips) et l'adapte à la situation courante.

1 INTRODUCTION

L'interaction entre des personnages virtuels et des utilisateurs en RV est un challenge. Le personnage virtuel est supposé agir comme le ferait un homme normal dans la même situation, ce qui pose de nombreux problèmes. Tout d'abord, le mouvement doit paraître naturel afin que l'utilisateur puisse interagir avec le personnage virtuel comme il le ferait avec un autre humain. Ensuite, le personnage virtuel doit calculer son mouvement très rapidement dans le but d'éviter des latences qui compromettraient la qualité de l'interaction. Le mouvement doit aussi être adapté à l'environnement, à la morphologie du personnage et aux actions du sujet. Enfin, comme l'humain virtuel ne peut pas prédire précisément les intentions de l'utilisateur, il est impossible d'utiliser des méthodes nécessitant une connaissance complète de la séquence à exécuter (comme c'est le cas pour les "cartes de déplacement" introduites par [4]).

*e-mail: npronost@irisa.fr
†e-mail : fmulton@irisa.fr
‡e-mail : blighli@gmail.com
§e-mail:gengwd@cs.zju.edu.cn
¶e-mail : rkulpa@irisa.fr
‖e-mail : Georges.Dumont@irisa.fr

La méthode proposée dans cet article a pour but d'animer des humains virtuels tout en résolvant les problèmes introduits ci-dessus. L'aspect naturel du mouvement est assuré par le recours à une base de données de mouvements capturés. Une approche classique consisterait alors à définir une très volumineuse base de données, généralement organisée sous la forme d'un "graphe de mouvements" [11] afin de gérer de nombreux cas de figure. Cependant, cela implique des temps de précalcul très importants et une phase manuelle fastidieuse. De plus, le graphe résultant est si important que de longs temps de calcul sont nécessaires pour trouver un chemin compatible avec la situation. Cette base de données est généralement dédiée à un squelette, ce qui conduit inévitablement à des calculs supplémentaires pour animer un personnage de taille différente. Par exemple, saisir un objet à 1 mètre de distance pour une grande personne conduit à un mouvement différent de celui qu'aurait fait quelqu'un de petit.

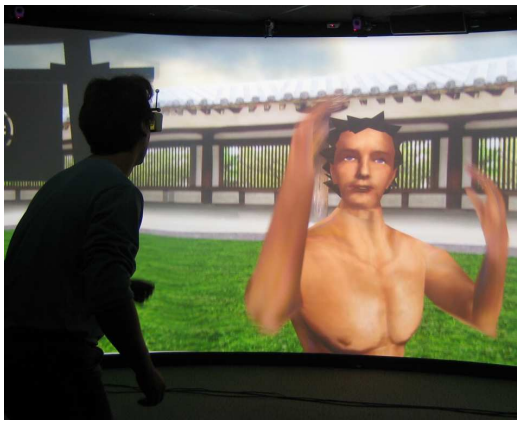

Figure 1: Interaction entre un sujet réel et un combattant de kung-fu virtuel.

Pour toutes ces raisons, l'approche présentée ci-dessus est inutilisable pour les applications interactives. Une alternative consisterait à adapter un unique mouvement (voire directement à générer les trajectoires articulaires). Cependant, le résultat pourrait sembler peu réaliste comparé à des mouvements capturés. Associer une petite base de données et des techniques d'adaptation de mouvements semble donc être un bon compromis pour résoudre ce type de problème en RV. Cet article décrit une méthode s'appuyant sur ce principe : moins de 20 mouvements dans la base de données, chaque élément de la base est associé à des informations nécessaires à son indexation (comme le type de mouvement, la partie du corps à contrôler, l'instant qui correspond à un événement pertinent et la morphologie de l'acteur). Nous proposons alors un algorithme de recherche qui sélectionne le mouvement les plus approprié à la situation, même s'il ne résout pas précisément toutes les contraintes. Cette méthode de recherche prend en compte la morphologie de l'acteur et du personnage à animer afin d'assurer que le mouve-

ment sélectionné est bien applicable à la situation dans le virtuel. Le mouvement sélectionné est adapté précisément, et en temps réel, au squelette du personnage et à l'environnement, et obéit aux ordres du superviseur.

Dans cet article, nous illustrons cette méthode sur l'exemple d'un combattant virtuel de kung-fu dont le rôle est de frapper avec les poings ou les pieds (suivant les ordres du superviseur) une cible localisée sur la tête d'un utilisateur réel. Ce dernier se déplace dans un environnement de 3m par 3m, dans une salle immersive, comme le montre la figure 1. Cette démonstration illustre toutes les contraintes listées ci-dessus. En effet, le moteur d'animation doit réagir à des événements imprévisibles (comme le déplacement de la tête de l'utilisateur ou les ordres transmis par le superviseur).

2 ETAT DE L'ART

Animer des humains virtuels à partir de mouvements capturés a été et continue à être largement exploré. Généralement, les méthodes proposées font l'hypothèse d'une connaissance complète des contraintes à respecter lors de l'exécution d'un mouvement. Les solutions proposées consistent alors à optimiser les trajectoires afin de résoudre un ensemble de contraintes à respecter. Dans cette famille d'approches, on trouve l'optimisation spatio-temporelle [3, 22] et les cartes de déplacement [4, 18]. La cinématique inverse est largement utilisée pour résoudre des contraintes définies à des instants donnés en considérant, par exemple, des priorités entre elles [15]. Il est possible de guider la solution vers une posture naturelle en précalculant le Jacobien pour un grand nombre de données capturées [8]. Le reste du traitement consiste alors à lisser les trajectoires résultantes au voisinage des contraintes [6] grâce à un processus itératif. Ceci nécessite un grand nombre de calculs et n'est donc pas compatible avec les contraintes des applications interactives.

Les graphes de mouvements [11, 1, 16, 2] ont été introduits dans le but de précalculer toutes les transitions possible a l'intérieur et entre des mouvements d'une base de données. Ceci permet de naviguer ensuite dans un graphe pour aller d'une configuration courante à celle qui est désirée en ne passant que par des postures capturées. Les transitions entre les configurations sont définies à partir d'une fonction de distance [25] mais doivent bien souvent être corrigées manuellement. Après le précalcul, le graphe résultant permet de chercher un chemin entre l'état courant du système et une configuration but. Si cette dernière n'existe pas dans le graphe, le système sélectionne celui qui est le plus proche dans la base de données. Ceci a été appliqué au contrôle d'un boxeur qui doit frapper une cible bougeant dans l'espace. Pour arriver à ce résultat des centaines de coups de poing ont dû être capturé afin d'échantillonner relativement précisément l'espace des cibles possibles autour du personnage [17]. Cependant, la taille importante du graphe conduit à de nombreux précalculs qui doivent être refaits si on ajoute un mouvement ou si le personnage a une taille différente. Plusieurs améliorations ont été proposées pour améliorer la performance de ces méthodes mais les précalculs ne permettent toujours pas de s'adapter rapidement à n'importe quel squelette en temps réel [7, 24, 10, 24].

Toutes les techniques présentées précédemment sont généralement liées à un squelette donné alors que plusieurs personnages différents sont à animer. La mise à l'échelle du mouvement [5] a été introduite pour adapter des trajectoires articulaires à la morphologie d'un personnage. Cette technique fait l'hypothèse que le principal problème provient de contraintes cinématiques à respecter. Les cartes de déplacement peuvent là-encore être utilisées pour résoudre ces problèmes. Cependant, elles ne s'appliquent pas aux environnements interactifs pour lesquels elle ne dispose pas de connaissances à l'avance sur le mouvement. Une représentation indépendante de la morphologie a été proposée afin de simplifier ce processus de mise à l'échelle [14]. Cette méthode permet d'animer en temps réel plusieurs centaines de personnages de taille différente tout en vérifiant ces contraintes cinématiques. En effet, cette représentation simplifie aussi le problème de résolution de contraintes cinématiques et cinétiques [13] en mixant des données Cartésiennes et angulaires. Cependant, ces méthodes se focalisent principalement sur des contrôles bas-niveau du personnage. Elles ne permettent donc pas de décider quel mouvement est le plus approprié en fonction de la situation à traiter. Dans l'exemple du combattant virtuel, quel est le mouvement le plus naturel et efficace pour frapper l'utilisateur avec ses poings et ses pieds?

Pour résoudre ce problème, une première idée pourrait être de coupler les graphes de mouvements avec des techniques d'adaptation en temps réel. Cependant, cela nécessiterait une très grosse base de données qui détériorerait les performances du système. L'alternative que nous proposons dans cet article consiste à utiliser des techniques de recherche de mouvements dédiées à de petites bases de données. Le résultat de la recherche, qui ne correspond certainement pas parfaitement à la situation, peut ensuite être adapté en temps réel. Dans cette méthode, la mise à l'échelle est un problème clé. En effet, un mouvement adapté à un petit personnage ne l'est peut-être plus pour un grand. La recherche doit donc prendre en compte la morphologie du personnage à animer. Dans notre exemple, l'utilisateur peut changer de personnage à tout moment. Le système doit s'adapter en temps réel pour trouver et modifier le mouvement en conséquence.

Li et Prabhakaran [19] ont proposé une structure arborescente pour indexer un mouvement avec des douzaines d'attributs. Des vecteurs de propriétés sont extraits pour l'indexation en utilisant une décomposition en valeurs singulières des matrices de mouvements. Lin [20] a proposé des propriétés qui représentent le mouvement de manière compacte à l'issue d'un précalcul. Liu et coll. [21] ont proposé un algorithme de recherche fondé sur le contenu 3D du mouvement. Ils partitionnent la base de données et construisent un arbre d'indexes fondé sur une description hiérarchique des mouvements. L'arbre d'indexes est utilisé comme classificateur pour déterminer le sous-ensemble qui contient les mouvements les plus prometteurs en fonction de la requête. Un algorithme de découpage en clusters fondé sur le plus proche voisin est utiliser pour décomposer la base de données en partitions et construire un arbre d'indexes. Deux mouvements sont dits similaires grâce à un algorithme de mise en correspondance élastique. Yu et coll. [27] ont mis en œuvre un processus qui permet à l'utilisateur de retrouver un mouvement via un langage d'annotations. Pour chaque clip de mouvement de la base de données, ils génèrent la séquence en notation de Laban correspondante. Une métrique de similarité pour séquences en notations de Laban est proposée pour résoudre des requêtes.

Comparé à des graphes de mouvements, et spécialement la méthode de Lee et Lee [17], nous utilisons une base de données limitée avec uniquement quelques mouvements car un algorithme d'adaptation permet d'obtenir une plus large variété de comportements. Nous assurons aussi de la mise à l'échelle automatique à la taille du personnage, ce qui est impossible avec l'approche de Lee et Lee. L'approche de Gleicher [5] nécessiterait une connaissance complète de la scène, ce qui est impossible en réalité virtuelle où l'utilisateur peut intervenir de manière imprévisible.

Comme la base de données est petite, avec des informations sémantiques pertinentes ajoutées comme clés, ce n'est pas nécessaire d'utiliser la méthode de Muller et coll. [23] fondée sur des *patterns*. Ces méthodes sont dédiées à des recherches automatiques dans une base de données inorganisée. Les captures de mouvements sont généralement traitées manuellement afin d'éliminer les occultations, filtrer les trajectoires, etc. A cette étape, annoter le fichier ne coûte pas beaucoup de temps.

Dans la suite de cet article, nous prenons l'exemple du combattant virtuel de kung-fu. Dans cet exemple, le combattant doit frap-

per une cible avec le poing et les pieds. La cible bouge avec la tête d'un utilisateur dont les mouvements sont capturés en temps réel dans une salle immersive. Ainsi, l'utilisateur peut se déplacer et s'accroupir à tout moment, sans aucune prédiction possible. Un superviseur ordonne au combattant de frapper avec les poings ou les pieds (à gauche ou à droite) en tapant des touches du clavier. Le combattant doit prendre toutes ces informations en considération, en temps réel, pour décider quel mouvement sélectionner dans la base de données (il dispose de plusieurs possibilités pour chaque coup).

3 MÉTHODES

3.1 Vue globale

Le processus global de cet exemple est décrit en figure 2.

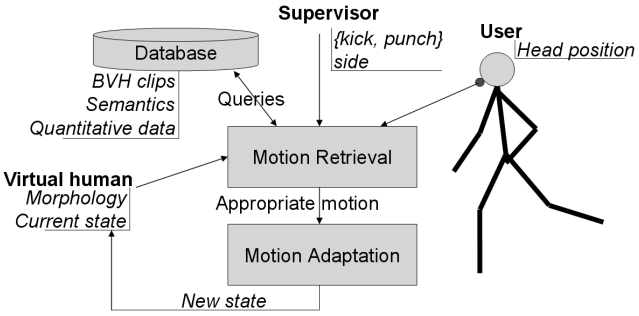

Figure 2: Vue globale de l'exemple du combattant virtuel.

Dans notre exemple, le mouvement de la tête du sujet est capturé en temps réel pour un système AR-Traking composé de 5 caméras infrarouges et de marqueurs réfléchissants. Avec ces données, la stéréovision et le point de vue peuvent être ajustés à chaque pas de temps. Pour le combattant virtuel, la cible est associée au milieu des marqueurs positionnés sur la tête.

Le moteur d'animation est composé de deux parties. La première concerne la recherche du meilleur mouvement dans une base de donées restreinte. La sélection s'effectue en fonction des paramètres suivants : configuration courante du combattant, position de la cible et l'ordre donné par le superviseur (coup de poing ou de pied à gauche ou à droite, se déplacer sinon).

Une fois que le mouvement correspondant est retrouvé, il doit être adapté au squelette du personnage à animer. C'est le rôle du second module. Le superviseur peut sélectionner à tout moment un personnage dans une bibliothèque. Des contraintes cinématiques (comme le contrôle des pieds sur le sol et la position de la cible à atteindre) sont résolues à chaque pas de temps. Le résultat est visualisé sur un écran de 9 mètres de long, en stéréovision.

La base de données de mouvements est composée de fichiers au format standard BVH incluant les trajectoires articulaires et la description du squelette de l'acteur. A chaque mouvement, quelques informations sont ajoutées via une interface graphique : sémantique (quel est le mouvement?), l'instant où un événement pertinent (comme la collision avec une cible lors d'une frappe) intervient...

Nous décrivons maintenant ces deux modules.

3.2 Recherche dans la base de données

Le processus général est présenté dans la figure 3.

Comme le personnage virtuel est certainement différent de l'acteur qui a effectué les captures de mouvements, quelques ajustements doivent être effectués. Premièrement, la position de la contrainte doit être mise à l'échelle pour correspondre à la taille du personnage. Par exemple, une contrainte placée à 1 mètre en avant du combattant n'a pas la même influence si ce dernier mesure 2m

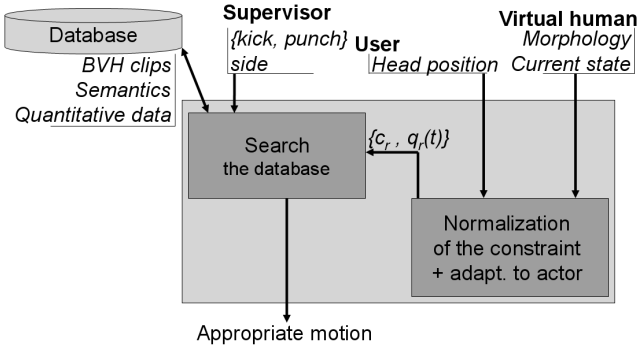

Figure 3: Vue générale du processus de recherche du mouvement adéquat dans la base de données.

ou 1.5m. Nous divisons la position relative de la contrainte c (exprimée dans le repère de la racine du combattant) par la taille du personnage $size_c$. Le résultat est une contrainte adimensionnelle qui est multipliée par la taille de l'acteur qui a effectué les captures de mouvements $size_a$:

$$c_r = c \times \frac{size_a}{size_c} \quad (1)$$

Deuxièmement, la posture courante du combattant $q(t)$ est remise à l'échelle de l'acteur appartenant à la base de données, fournissant ainsi la valeur q_r. En conséquence, la distance entre q_r et une posture du clip est fondée sur le même squelette. La mise à l'échelle est effectuée grâce à la méthode décrite en section 3.3.1 et utilisée pour le module *Motion Adaptation* (voir sous-section 3.3). Ainsi, les entrées de l'algorithme de recherche (appelé *Search* en figure 3) sont c_r et q_r.

Rechercher le mouvement qui correspond le mieux à la fois à la contrainte et à la posture courante pourrait engendrer de nombreux temps de calcul. Pour accélérer le processus, nous proposons d'organiser la base de données d'une manière à optimiser les calculs. Premièrement, les mouvements sont sélectionnés dans une base de données plus vaste afin de gérer plusieurs types de coups de poing et de pied. Deuxièmement, la base de données résultante est organisée en clusters afin d'accélérer la recherche d'un mouvement qui tienne compte de la posture courante du personnage et de la contrainte à respecter.

3.2.1 Conception d'une base de données restreinte

Comme la qualité des mouvements animés dépend du contenu de la base de données, nous avons accordé un soin particulier à sélectionner un petit ensemble de mouvements de combats qui correspondent aux besoins de la démonstration. Cette sélection est précalculée au sein d'une base de données enregistrée au format standard BVH. Grâce à un ensemble d'outils développé dans notre laboratoire, l'utilisateur peut extraire les sous-séquences les plus importantes de la base de données (comme celles où le personnage frappe avec ses poings ou ses pieds). Pour chaque sous-séquence, l'utilisateur indique le nom du segment utilisé pour le coup ainsi que la sémantique associée (coups de poing ou de pied). Le système calcule automatiquement la trajectoire du point servant au coup. Pour chaque mouvement, le système calcule aussi automatiquement la boîte englobante de cette trajectoire, comme le montre la figure 4. L'algorithme *Search* de recherche est supposé éliminer les mouvements dont la boîte englobante ne contient pas la contrainte c_r.

3.2.2 Décomposition en clusters

Comme le processus complet doit s'exécuter en temps réel, le temps de calcul pour rechercher le mouvement est un point important.

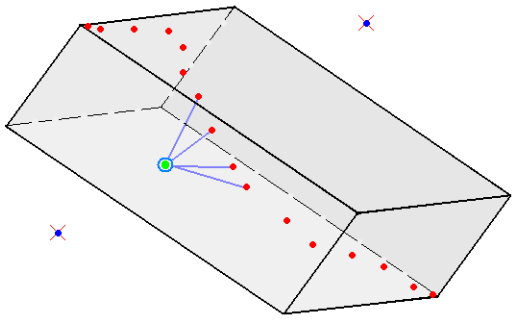

Figure 4: Boîte englobante de la trajectoire du point de frappe, modélisée par des points rouges (comme la trajectoire du poignet). La contrainte c_r est modélisée par des points verts.

Nous proposons d'organiser la base de données en clusters lors d'une phase de précalcul. Le but est de grouper les postures de plusieurs mouvements qui sont compatibles entre elles. Ainsi, la méthode devrait regrouper les postures $m_i(t)$ et $m_j(t')$ du mouvement m_i et m_j si la position du point de frappe est similaire. Par exemple, si G est le cluster contenant $m_i(t)$ et $m_j(t')$, il sera défini par :

$$G = \{m_i(t), m_j(t')\} \tag{2}$$

Dans le reste de cet article, le i^{eme} élément de G est noté $Pose_i^G$.

Afin de résoudre ce problème, nous choisissons l'algorithme K-mean. Le but de cet algorithme est de regrouper les postures de différents mouvements en clusters afin de minimiser la variance intra-clusters. L'algorithme commence par partitionner aléatoirement les postures les postures en k ensembles. Alors, il calcule la valeur moyenne des postures pour chaque cluster (la valeur moyenne est aussi appelée centrode $\overline{Pose^G}$). L'algorithme construit une nouvelle partition en associant chaque posture avec le centroide le plus proche. alors, les centroides sont recalculés pour les clusters résultants. La méthode (construire de nouvelles partitions et calculer des centroides) est répété jusqu'à convergence. La convergence est obtenue quand une posture arrête de passer d'un cluster à l'autre. Un autre critère de convergence est que les centroides arrêtent de changer après une itération [9].

Afin de savoir si deux postures sont similaires, une distance doit être définie. Comme les postures d'un clip ont une grande variété d'orientation et de position, définir une telle distance est difficile. Une solution consiste à déplacer la racine du personnage et le réorienter selon un axe unique. Une meilleure solution proposée par Kovar et coll. [12] consiste à minimiser la distance entre les deux postures en faisant varier les paramètres d'alignement (position et orientation) :

$$\min_{\theta,x_0,z_0} \sum_i w_i \| p_i - T_{\theta,x_0,z_0} P_i' \|^2$$

où

$$\theta = \arctan \frac{\sum_i w_i (x_i z_i' - x_i' z_i) - \frac{1}{\sum_i w_i}(\bar{x}\bar{z}' - \bar{x}'\bar{z})}{\sum_i w_i (x_i x_i' - z_i z_i') - \frac{1}{\sum_i w_i}(\bar{x}\bar{x}' - \bar{z}\bar{z}')} \tag{3}$$

et

$$x_0 = \frac{1}{\sum_i w_i}(\bar{x} - \bar{x}'\cos(\theta) - \bar{z}'\cos\theta) \tag{4}$$

$$z_0 = \frac{1}{\sum_i w_i}(\bar{z} + \bar{x}'\sin(\theta) - \bar{z}'\sin\theta) \tag{5}$$

Dans cette équation, w_i est un poids associé au point p_i de la trajectoire et $T_{\theta,x_0,z_0} P_i'$ appartient à celle qui est transformée par translation (x_0, y_0) et rotation d'un angle θ.

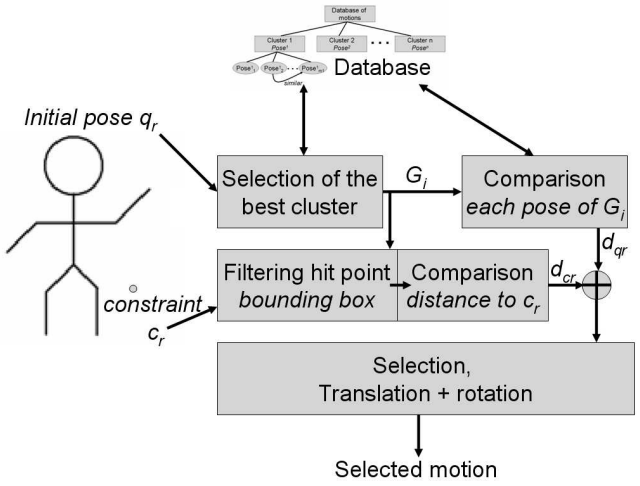

Figure 5: Vue générale de l'algorithme de recherche *Search*.

3.2.3 Algorithme de recherche

Après les calculs de clusters, la base de données et organisée comme un ensemble de groupes G associés à une posture typique (ou centroide) $\overline{Pose^G}$. Le principe général de l'algorithme de recherche est décrit en figure 5. Il consiste à sélectionner un cluster qui correspond le mieux à la posture initiale q_r et en associant une distance d_{q_r} à chaque posture de ce cluster. Pour cela, tous les centroides $\overline{Pose^{G\in\{clusters\}}}$ du cluster G sont comparés avec la posture initiale q_r. Soit G_i le cluster ayant une posture moyenne $\overline{Pose^{G_i}}$ correspondant le mieux à q_r. G_i est alors sélectionné pour comparer toutes les postures correspondantes $Pose_{j\in G_i}^{G_i}$ et q_r. Les postures sont représentées comme des feuilles d'un arbre, comme le montre la figure 6. Toutes les feuilles sont associées à un ensemble de postures similaires $similar(Pose_j^{G_i})$. Celles-ci sont des voisins dans la structure de données, ce qui permet de rapidement pointer sur des postures similaires. Par exemple, en figure 6, la posture $Pose_2^1$ du cluster G_1 est sélectionnée par l'algorithme. La posture similaire est $Pose_{m_1}^1$.

Même si $Pose_i^{G_1}$ semble être le meilleur candidat selon le critère de distance d_{q_r} entre $Pose_i^{G_1}$ et q_r, il n'est peut-être pas optimal pour d'autres critères. Par exemple, cette posture pourrait correspondre au début d'un coup de poing ou de pied mais pas à l'instant du contact. Dans ce cas, les postures similaires sont référencées par $similar(Pose_i^{G_1})$ et peuvent être rapidement analysées afin de décider si elles correspondent mieux à l'instant du coup. Supposons que l'une de ces postures similaires correspond mieux à un coup que $Pose_j^{G_i}$. Cette dernière est finalement sélectionnée même si sa distance à la contrainte est plus mauvaise que celle de $Pose_j^{G_i}$ afin d'éviter des mouvements peu réalistes.

Cette sélection prend uniquement en compte la posture initiale mais résoudre la contrainte c_r est aussi un problème important. Les candidats résultants doivent être filtré afin d'éliminer ceux dont la trajectoire du point de frappe n'est pas compatible avec c_r. Cette tâche est effectuée en éliminant les mouvements dont la boîte englobante (voir figure 4) ne contient pas q_r. Chaque posture restant est associée avec la distance minimale d_{c_r} entre la trajectoire et le point de frappe c_r. La posture (et le mouvement correspondant) qui est finalement sélectionnée par cet algorithme est celle qui minimise la somme pondérée de deux distances :

$$d = w_{q_r} \times d_{q_r} + w_{c_r} \times d_{c_r} \tag{6}$$

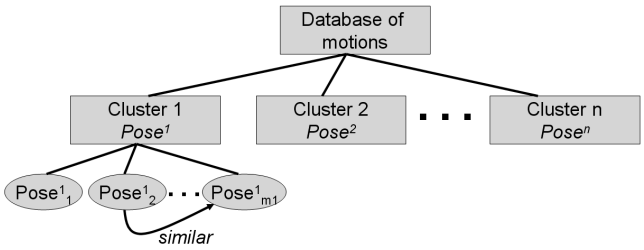

Figure 6: Structure de données hiérarchique utilisée pour organisée les mouvements.

où w_{c_r} et w_{q_r} sont initialisée au départ de l'algorithme.

Le mouvement associé à la plus petite distance d est alors tourné et déplacé pour correspondre le mieux possible à la posture initiale q_r. Le mouvement résultant est fourni au module d'adaptation *motion adaptation*.

3.3 Adaptation de mouvements

Le but de ce module est d'adapter le mouvement sélectionné M_s à la situation courante. Comme le montre la figure 7, ceci implique de :

- synchroniser et mélanger M_s avec le mouvement en cours,
- mettre à l'échelle M_s pour l'adapter au squelette du combattant virtuel,
- et résoudre les contraintes cinématiques (par exemple, guider le poing vers la cible et corriger les contacts des pieds avec le sol).

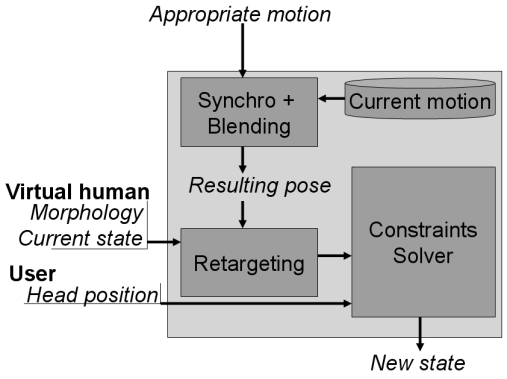

Figure 7: Principe général du processus d'adaptation de mouvements.

Mettre à l'échelle consiste principalement à multiplier des données adimensionnelles par les dimensions des différentes partie du corps du personnage. Une autre partie concerne la correction des positions des pieds pour éviter des effets de glissement, de vol ou de pénétration dans le sol (voir le paragraphe 3.3.1). La posture résultante est compatible avec le squelette et le sol mais ne vérifie toujours pas les autres contraintes. Le paragraphe 3.3.2 explique comment résoudre ces contraintes très rapidement.

3.3.1 Mise à l'échelle

La mise à l'échelle est l'un des points clé de cet article. En effet, la morphologie du personnage intervient lors de la recherche dans la base de données et pour adapter le mouvement sélectionné au combattant virtuel. Pour résoudre ce type de problème, la méthode la plus communément utilisée consiste à appliquer les trajectoires articulaires du personnage #1 au personnage #2 tout en corrigeant les postures en fonction de contraintes géométriques [5]. Ceci évite les artéfacts classiquement rencontrés au niveau des pieds. Cependant, cette correction est très coûteuse en temps de calcul et implique une connaissance complète du mouvement à l'avance pour éviter les discontinuités.

Une alternative consiste à utiliser une représentation indépendante de la morphologie, comme cela a été présenté dans [14]. Comme la mise à l'échelle implique de résoudre des contraintes géométriques, cette représentation s'appuie sur des données Cartésiennes qui évitent le recours à la cinématique inverse. Ainsi, comme le montre la figure 8, le bras est modélisé par une ligne reliant l'épaule et le poignet (notée l_i pour le membre i), ainsi qu'un angle α_i pour localiser la direction dans laquelle le coude se place (sans donner précisément sa position qui dépend des longueurs des bras et avant-bras). La position relative l_i est divisée par la taille totale du membre i (ex. : bras et avant-bras) conduisant à des données adimensionnelles.

De la même manière, toutes les parties intermédiaires du squelette sont modélisées par la positon relative du point distal dans le repère proximal, divisé par la taille du segment. Le torse est modélisé par une spline qui peut être échantillonnée en autant de segments que le squelette final en comporte. Soit q_{adim} la structure de données résultante(contenant des données adimensionnelles) :

$$q_{adim} = \left\{ \begin{array}{l} \frac{root}{size(legs)}, spline, \frac{clav_{left/right}}{size(clav_{left/right})}, \\ \frac{hip_{left/right}}{size(hip_{left/right})}, \\ l_{la,ra,ll,rl}, \alpha_{la,ra,ll,rl}, \\ \frac{fingers_{left/right}}{size(fingers_{left/right})}, \\ \frac{foot_{left/right}}{size(foot_{left/right})}, \\ \frac{head}{size(head)} \end{array} \right\} \quad (7)$$

où la, ra, ll, rl sont respectivement le bras gauche, droit, la jambe gauche et droite.

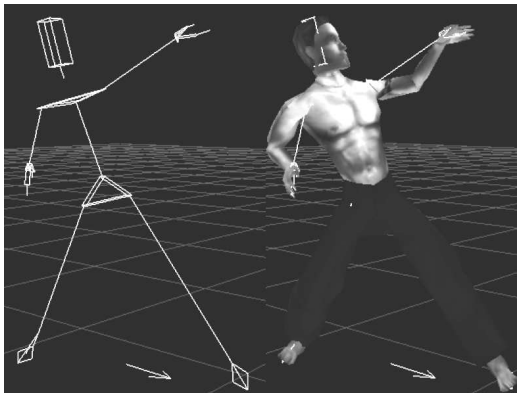

Figure 8: Représentation indépendante de la morphologie (à gauche) et sa relation avec la posture du personnage 3D (à droite).

Pour la visualisation, le passage à une représentation classique (utilisant des angles aux articulations) est obtenu en trois principales étapes. Premièrement, les données adimensionnelles sont multipliées par les dimensions des segments du personnage à animer. Deuxièmement, si des contraintes géométriques sur les pieds ne sont pas vérifiées, ces derniers peuvent être directement placés aux bonnes positions. Considérons qu'un pied n'est pas en contact avec le sol ou glisse. Comme les données stockées sont

Cartésiennes, il suffit simplement de décaler le pied à la position souhaitée, sans changer les autres informations. La posture résultante (mise à l'échelle et corrigée au niveau des pieds) est appelée q_{scaled}.

Troisièmement, les angles aux articulations sont calculés pour être envoyés à la visualisation. Ce processus s'effectue en partant des pieds vers le reste du corps, en recalculant les positions absolues de chaque articulation (le bassin est placé par rapport aux pieds, etc.). Pour chaque articulation, un repère local est défini à partir de ces informations. Ce repère local permet de calculer les angles entre deux segments adjacents. Pour les membres, la position du coude ou du genou doit préalablement être retrouvée. Cette articulation intermédiaire est placée à l'intersection de deux sphères (chacune centrée sur le point distal et proximal du membre) et du demi-plan lié à l'angle α_i.

En utilisant cette méthode, on atteint des performances de calcul très intéressantes : moins de $88\mu s$ sur un ordinateur portable du commerce. Pour plus de détails sur cette méthode, le lecteur est invité à consulter [14].

3.3.2 Résolution de contraintes

La méthode décrite ci-dessus a pour but d'adapter une posture à un personnage dont les dimensions sont différentes de celles de l'acteur. Cela revient principalement à résoudre des contraintes géométriques sur les pieds. Cependant, d'autres contraintes doivent aussi être vérifiées : principalement toucher la cible avec la main ou le pied. Rappelons que la base de données est petite comparé à des graphes de mouvements ou d'autres approches classiques. En conséquence, le mouvement sélectionné n'est certainement pas adapté précisément aux contraintes et doit donc être modifié. Cette modification doit être rapide afin d'éviter d'augmenter la taille de la base de données qui conduirait à d'importants temps de calculs pour retrouver le mouvement le plus adapté.

Résoudre des contraintes cinématiques est généralement effectué via des cartes de déplacements [4]. Le principal avantage de cette méthode est d'assurer la continuité du mouvement résultant tout en vérifiant les contraintes. Cependant, cela nécessite d'importants temps de calculs, ce qui est incompatible avec les applications interactives? Cependant, cela implique trop de temps de calculs. Dans notre cas, quand le superviseur donne un ordre, le système doit réagir en moins de 1/30s. Nous proposons donc une résolution image par image en se fondant sur des contraintes exprimées de manière continue. Pour le mouvement sélectionné M, le temps entre l'image courante t_0 et celle correspondant à l'instant de contact avec la cible t_c est connu. A t_c, la contrainte correspond au vecteur $d(t_c)$ entre $M(t_c)$ et la cible X. A t_0, cette distance est $d(t_0) = 0$. Entre les deux, $d(t)$ est une interpolation entre $d(t_0)$ et $d(t_c)$. Ce problème est décrit en figure 9.

Figure 9: Résolution de contraintes cinématiques (personnage gris clair : avec adaptation; personnage normal : pas d'adaptation.

A n'importe quel instant t, nous avons donc à adapter les angles articulaires du personnage afin de guider le poignet ou le pied vers une nouvelle position $p(t)+d(t)$, où $p(t)$ est la position initiale sans aucune adaptation. Pour résoudre ce problème, la cinématique inverse avec Jacobien et des contraintes par priorités est généralement utilisée [15]. Cependant, cela conduit à de nombreux calculs qui peuvent être incompatible avec des applications interactives. Encore une fois, nous avons choisi la méthode proposée dans [14]. Ceci consiste à utiliser q_{scaled} plutôt que des angles articulaires afin de résoudre ce problème. Ces données q_{scaled} sont des positions relatives exprimées dans un repère Cartésien. Ainsi, modifier la position d'une extrémité est immédiat si la contrainte est atteignable par le membre. En effet, cela consiste à changer le vecteur relatif entre la position de l'extrémité (comme le poignet) et du point proximal (comme l'épaule). Nous faisons l'hypothèse que α_i est inchangé afin de préserver le style du mouvement original.

Cependant, dans un grand nombre de cas, la contrainte n'est pas atteignable en n'utilisant que le membre correspondant. Dans ce cas, nous utilisons un algorithme itératif dérivé du *Cyclic Coordinate Descent* [26] et illustré en figure 10. Soit v_i le vecteur reliant le point contraint (comme le poignet) et le point proximal du segment i. Soit $\widehat{v_i}$ le vecteur reliant la contrainte (la tête de l'utilisateur) et le point proximal du segment i. La méthode tente d'aligner v_i et $\widehat{v_i}$ pour toutes les parties du corps, les unes après les autres. Le segment distal est essayé en premier, puis les autres jusqu'à la racine du personnage, comme le montre la figure 10 pour le poignet. A la fin, si la contrainte n'est toujours pas satisfaite, une nouvelle itération est initiée. Chaque itération est très rapide car l'alignement revient à travailler sur une chaîne composée d'un ou de deux segments.

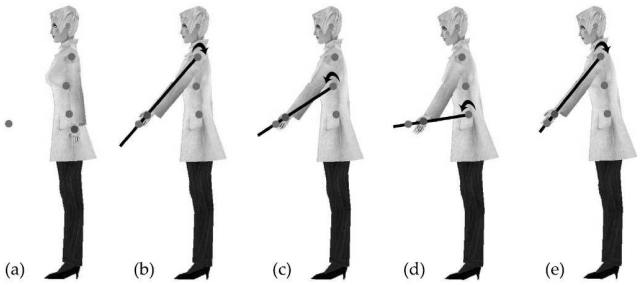

Figure 10: Processus itératif utilisé pour résoudre les contraintes cinématiques a) posture originale b) utilisation du bras c) utilisation de la clavicule d) utilisation de l'abdomen e) nouvelle itération avec le bras à nouveau.

Plus les contraintes sont éloignées, plus il faut de temps de calcul. Si uniquement un membre est impliqué dans le calcul, cette méthode ne demande que $88\mu s$, comme pour la mise à l'échelle. Pour des contraintes atteignables, mais nécessitant 3 parties différentes du corps (bras, clavicule et abdomen), les temps de calcul sont de l'ordre de $515\mu s$. Pour des contraintes inatteignables, au pire, le temps de calcul est de $1240\mu s$ ce qui reste compatible avec une animation à 30Hz.

3.4 Protocole expérimental

Plusieurs personnages virtuels, avec des dimensions différentes, ont été définis avec Avatar Studio (produit de Canal NuMedia). Au début de l'expérimentation, le superviseur sélectionne l'un de ces personnages mais il peut changer de combattant à n'importe quel instant en utilisant le clavier. Pas seulement la taille mais aussi les proportions entre les segments du corps sont différents d'un personnage l'autre.

Nous avons capturé le mouvement d'un maître de kung-fu. Ce dernier devait effectuer :

- 4 coups de pieds différents de chaque côté dans le but d'avoir différents styles et cibles dans l'espace (coup haut, bas, médian et un autre, libre),
- 4 coups de poing pour chaque côté.

Chaque mouvement était encodé au format standard BVH associé à quelques informations nécessaires pour le caractériser. Ainsi, 16 petites séquences (limitées un seul coup par s´quence) ont été enregistrées dans la base de données, ce qui est bien inférieur à ce que l'on trouve dans les graphes de mouvements.

Le mouvement sélectionné dans le module de recherche est mélangé avec celui qui est en cours d'animation en ajustant des

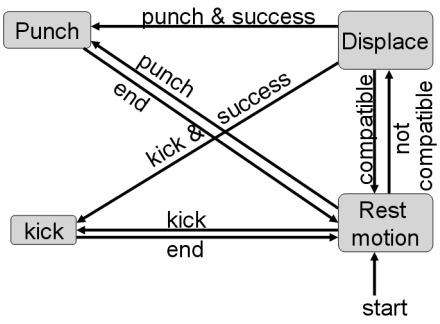

Figure 11: Automate décrivant le comportement simplifié du combattant de kung-fu.

poids de manière continue. Le module d'adaptation est alors appliqué afin d'assurer que la cible est bien atteinte par le pied ou la main, ce qui n'est généralement pas le cas autrement. Quand un coup de pied ou de poing est en cours d'exécution, les nouveaux ordres du superviseur ne sont pas pris en compte. Ainsi, les coups de poing et de pied sont uniquement mélangés avec des déplacements ou le mouvement de repos. Ainsi, pour des coups de pieds, l'adaptation doit synchroniser les contacts d'appui des deux mouvements (coup de pied et repos) afin d'éviter des enchaînements peu réalistes. Par exemple, donner un coup de pied à gauche alors que l'appui est du même côté conduit inévitablement à des mouvements irréalistes.

Quand l'utilisateur entre dans la pièce, les marqueurs sont positionnés sur sa tête. La caméra virtuelle est déplacée à la position de la tête de l'utilisateur. La vision stéréoscopique est adaptée aux déplacements de la tête à chaque image. Au début, le combattant virtuel est placé deux mètres en avant du sujet et est animé avec le mouvement de repos. Il commence ensuite à se déplacer en direction de l'utilisateur tout en restant dans une zone compatible avec des coups de pied ou de poing contre l'utilisateur. Le comportement du combattant est schématisé en figure 11.

Chaque fois que le superviseur sélectionne un coup de pied ou de poing, la position de la tête de l'utilisateur est enregistrée (soit *target* cette position). Après cela, si l'utilisateur bouge, *target* n'est pas mis à jour si bien que ce dernier peut éviter les coups. A l'inverse, s'il ne bouge pas, il est possible de vérifier que le coup arrive bien à destination.

4 Résultats

Nous avons demandé à un sujet de participer à cette étude. Ce dernier est équipé avec des lunettes de stéréovision (lunettes Crystal Eyes de stéréovision active). 5 marqueurs réfléchissants composent un corps rigide qui est attaché sur les lunettes. Le sujet prend place dans lenvironnement immersif afin dinteragir avec des combattants virtuels (comme présenté en figure 1).

Quand lutilisateur (ici, le sujet) se déplace à lintérieur de lenvironnement, le combattant virtuel se déplace également afin de le conserver à une distance atteignable par des coups de pieds et de poings. Dans la figure 12, la tête de lutilisateur est représentée par une sphère bleue et le coup de pied droit (sélectionné dans la base de données) est adapté afin datteindre la position de la sphère.

Dans lenvironnement virtuel, les paramètres du point de vue sont modifiés en temps réel en fonction du déplacement de la tête de lutilisateur. Ainsi, pour chaque coup, limpact est toujours situé au milieu de lécran (correspondant à la position du point de vue dans le monde virtuel). Comme souhaité, ce positionnement est correct pour lutilisateur, mais il peut être vu comme incorrect pour une personne dans une position différente. La partie gauche de la figure 13 montre un coup de pied droit qui atteint le milieu de lécran, mais

Figure 12: Adaptation de différents coups sélectionnés dans la base de données. La posture modifiée est affichée en bleu clair et la cible est représentée par une sphère bleue.

le point de vue de lobservateur est loin de lutilisateur. Dans ce cas, du point de vue de lobservateur, le coup de pied semble rater la cible. La partie droite de cette figure montre ce que voit réellement lutilisateur: le pied est au milieu de lécran et est crédible pour lui.

Figure 13: Vue extérieure de linteraction entre lutilisateur et le combattant virtuel (Gauche). Le même type de scène vue par lutilisateur (Droite).

Nous avons évalués les performances de notre méthode sur un ordinateur dont la configuration était la suivante: Pentium 4 3.6GHz, 2Go de mémoire. Lalgorithme de recherche était exécuté sur un processus séparé du reste de la méthode. La latence entre lordre du superviseur et le début de la recherche est approximativement de 3ms. Dans le processus de recherche, le temps entre le début de lalgorithme de recherche et lobtention du résultat du mouvement sélectionné est presque de 0.28 sec. Comme nous utilisons deux processus séparés, le système continue danimer lhumain virtuel pendant la sélection du mouvement. Ce mouvement est fourni à travers un fichier BVH écrit sur le disque dur. Ce fichier est alors chargé et convertit par une représentation indépendante de la morphologie afin de ladapter au personnage virtuel. La durée totale de ce traitement est denviron 40ms ce qui est excessif pour une animation à 25Hz. Cependant, la plus grande partie de cette durée est consacrée à la lecture du fichier depuis le disque. Elle pourrait donc être facilement diminuée si nous utilisons la mémoire au lieu du disque dur. Rappelons ici que ladaptation morphologique couplée à la gestion des contraintes ne prennent que 88μs par personnage.

5 Conclusion

Nous avons présenté une méthode danimation dhumains virtuels qui est capable dinteragir avec des sujets virtuels en temps réel. Lintérêt principal de cette approche est dassocier deux techniques utilisées en animation dhumains virtuels: la recherche dans une base de données et ladaptation de mouvement. Cette association soulève plusieurs difficultés telles que fouiller une base de données de mouvements capturés sur des acteurs alors que les humains virtuels et les contraintes sont différents. Comparé aux graphes de mouvements classiquement utilisés pour résoudre ce type de problème, notre méthode est capable de gérer des personnages de tailles différentes en temps réel. La différence de morphologie est directement prise en compte dans le processus de recherche de

mouvement, ce qui nous permet de trouver la réaction la plus appropriée en fonction de contraintes temps réel telle que des coups de pied et des coups de poings sur des cibles mouvantes.

Les temps de calculs sont assez bas par deux raisons. Premièrement, la taille de la base de données est petite, en particulier comparée aux méthodes basées sur les graphes de mouvements. En conséquence, effectuer une recherche dans cette base de données nest pas couteux. De plus, nous proposons, une organisation de la base de données qui accélère la recherche en prenant en compte diverses contraintes: la posture courante du personnage virtuel et la position de la cible. Le manque de données dans la base est compensé par lalgorithme dadaptation de mouvement. Ensuite, ladaptation à la morphologie et à lenvironnement est basée sur une représentation indépendante de la morphologie qui permet daccélérer la résolution des contraintes cinématiques. Les performances indiquées dans cet article peuvent être considérablement améliorées en utilisant la mémoire au lieu du disque dur pour transférer les mouvements depuis les deux processus (une écriture pour lalgorithme de recherche et une lecture pour le reste).

Dans cet article, nous nous sommes intéressés à lexemple du combattant de kung-fu mais ce travail peut être utilisé dans diverses applications faisant interagir des humains virtuels et réels dans un environnement virtuel. Grâce au faible coût de calcul, ce travail peut être utilisé pour permettre des tâches collaboratives entre humains réels et virtuels, comme par exemple dans une usine numérique. Dans ce cas, le modèle comportemental simple basé une machine à états doit être amélioré. Nous pouvons imaginer dassocier ce module à des modèles comportementaux afin de gérer des situations complexes faisant intervenir plusieurs humains virtuels.

Comme perspectives, nous souhaitons accroître les capacités de lhumain virtuel en lui fournissant de nombreux comportements différents. Lidée est de donner plus dautonomie aux humains virtuels sans avoir à gérer de très grande base de données qui sont difficilement manipulables en temps réel. De la même manière, nous souhaitons évaluer plus précisément comment les temps de calcul sont affectés lorsque la base de données est remplie avec des mouvements capturés sur de nombreux acteurs différents. Plus généralement, nous voulons valider cette approche en comparant des comportements réels à ceux que nous obtenons. Comme nous proposons de travailler avec un ensemble limité de mouvements, le choix de ces quelques mouvements est certainement très important. Nous voulons maintenant expérimenter comment sélectionner automatiquement un ensemble minimal de mouvements parmi une grande base de données afin de produire des comportements réalistes avec peu de temps de calcul.

Une autre piste intéressante serait de gérer des interactions bidirectionnelles dans lesquelles lhumain virtuel pourrait réagir par exemple à des coups de poids et des coups de pieds de lutilisateur. Dans une usine numérique, lutilisateur pourrait donner des outils à des humains virtuels. Ces derniers pourraient alors attraper des outils et lutiliser pour effectuer des tâches complexes en interaction étroite avec lutilisateur.

REFERENCES

[1] O. Arikan and D. Forsyth. Interactive motion generation from examples. *Proceedings of SIGGRAPH 2002*, 2002.

[2] M. Choi, J. Lee, and S. Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22(2):182–203, 2003.

[3] M. Cohen. Interactive spacetime control for animation. *Proceedings of ACM SIGGRAPH '92*, .26:293–302, July 1992. Chicago, Illinois.

[4] M. Gleicher. Motion editing with spacetime constraints. In *In Proceedings of Symposium on Interactive 3D Graphics*, pages 139–148, 1997.

[5] M. Gleicher. Retargetting motion to new characters. In *Proc. of ACM SIGGRAPH*, pages 33–42, July 1998.

[6] M. Gleicher. Motion path editing. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 195–202. ACM Press, 2001.

[7] M. Gleicher, H. Shin, L. Kovar, and A. Jepsen. Snap together motion: Assembling run-time animation. In *Proceedings of Symposium on Interactive 3D Graphics*, 2003.

[8] K. Grochow, S. Martin, A. Hertzmann, and Z. Popovic. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522–531, 2004.

[9] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[10] L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics*, 23(3):559–568, 2004.

[11] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, 2002.

[12] L. Kovar, M. Gleicher, and J. Schreiner. Footstake cleanup for motion capture. *ACM Siggraph Symposium on Computer Animation 2002*, 2002.

[13] R. Kulpa and F. Multon. fast inverse kinematics and kinetics solver for human-like figures. In *Proceedings of IEEE Humanoids*, pages 38–43, Tsukuba, Japan, december 2005.

[14] R. Kulpa, F. Multon, and B. Arnaldi. Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum, Eurographics 2005 special issue*, 24(3):343–352, 2005.

[15] B. LeCallennec and R. Boulic. Interactive motion deformation with prioritized constraints. In D. P. R. Boulic, editor, *Proceedings of ACM/Eurographics SCA*, pages 163–171, Grenoble, France, august 2004.

[16] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 491–500. ACM Press, 2002.

[17] J. Lee and K. Lee. Precomputing avatar behavior from human motion data. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 79–87, Grenoble, France, August 2004.

[18] J. Lee and S. Shin. A hierarchical approach to interactive motion editing for human-like figures. *Proceedings of ACM SIGGRAPH 99*, pages 39–48, Aug. 1999.

[19] Q. Li, W. Geng, T. Yu, X. Shen, N. Lau, and G. Yu. Motionmaster: Authoring and choreographing kung-fu motions by sketch drawings. In *Proceedings of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 233–241, 2006.

[20] Y. Lin. Efficient human motion retrieval in large databases. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer Graphics and interactive techniques in Australia and Southeast Asia*, pages 31–37, Kuala Lumpur, Malaysia, 2006.

[21] F. Liu, Y. Zhuang, F. Wu, and Y. Pan. 3d motion retrieval with motion index tree. *Comput. Vis. Image Underst.*, 92(2-3):265–284, 2003.

[22] Z. Liu, S. Gorther, and M. Cohen. Hierarchical spacetime control. *Proceedings of ACM SIGGRAPH '94*, pages 35–42, July 1994. Orlando, Floride.

[23] M. Muller, T. Rober, and M. Clausen. Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics*, 24(3):677–685, 2005.

[24] H. Shin and H. Ko. Fat graphs: Constructing an interactive character with continuous control. In *Proc of Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, 2006.

[25] J. Wang and B. Bodenheimer. An evaluation of a cost metric for selecting transitions between motion segments. *Eurographics/SIGGRAPH Symposium on Computer Animation 2003*, pages 232–238, 2003.

[26] L.-C. T. Wang and C. C. Chen. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Trans. On Robotics and Applications*, 7(4):489–499, August 1991.

[27] T. Yu, X. Shen, Q. Li, and W. Geng. Motion retrieval based on movement notation language. *Computer Animation and Virtual Worlds*, 16(3-4):273–282, 2005.