Mesures et modèles pour la capture de mouvement

Lionel Reveret

▶ **To cite this version:**

Lionel Reveret. Mesures et modèles pour la capture de mouvement. Traitement du signal et de l'image [eess.SP]. Institut National Polytechnique de Grenoble - INPG, 2014. ⟨tel-01064134⟩

UNIVERSITÉ DE GRENOBLE

HABILITATION À DIRIGER DES RECHERCHES

Spécialité : **Informatique**

Présentée par

Lionel Revéret

Mesures et Modèles pour la Capture de Mouvement

soutenue publiquement le **16 mai 2014**,
devant le jury composé de :

Jean-Paul Laumond
Directeur de Recherche CNRS, LAAS, Rapporteur
Ronan Boulic
Maître d'Enseignement et de Recherche, EPFL, Rapporteur
Karan Sher Singh
Professeur, Université de Toronto, Rapporteur
Michael Gleicher
Professeur, Université du Wisconsin Madison, Examinateur
Sylvie Gibet
Professeure, Université de Bretagne, Examinatrice
Marie-Paule Cani
Professeure, Grenoble INP, Examinatrice

RÉSUMÉ

Il est beaucoup plus fréquent d'entendre parler de capture que de mesure de mouvement. On peut y voir l'intuition que derrière le mot mouvement se conçoit un phénomène plus complexe que la donnée de marqueurs qui en constitue aujourd'hui la norme d'instrumentation rigoureuse. Si le marqueur est quantifiable, le mouvement conserve une qualité supplémentaire à explorer. Je retrace ainsi ici les travaux de recherche que j'ai encadrés ces dernières années sur cette notion de capture de mouvement, à travers les outils scientifiques que sont *la mesure* et *le modèle*. Mes activités ont été initialement dédiées à l'animation 3D, puis se sont progressivement tournées vers des enjeux liés à l'anatomie. Les contributions en animation 3D ont d'abord porté sur la recherche d'espaces paramétriques optimaux pour mesurer et générer le mouvement articulé. L'optimalité est à prendre ici au sens de la recherche d'une réduction de dimensions qui préserve au mieux la qualité du mouvement. Elle s'est déclinée autour d'applications pour l'analyse vidéo du mouvement, en particuliers animal, sur la compression de données de mouvement articulé et l'édition de pose de personnage 3D. Ces différents thèmes ce sont structurés autour de modèles d'analyse statistique multidimensionnelle appris des différentes sources données, vidéo ou articulaires, conduisant à un paramétrage de haut niveau du mouvement. J'ai ensuite abordé différents aspects de l'intégration de données réelles dans des modèles d'animation physiques. Le mouvement animal a été étudié à travers une simulation de quadrupèdes dont les paramètres ont été optimisés par rapport à des données "terrain". Le mouvement humain a quant à lui été modélisé à travers le développement d'une formulation en mécanique Lagrangienne des paramètres de haut niveau identifiés précédemment. Une mesure des aspects dynamiques a été menée pour les situations de contacts multiples avec une application d'estimation de forces directement à partir de la cinématique. Une grande partie de mon activité de recherche a aussi porté sur le développement de systèmes expérimentaux pour le petit animal de laboratoire. Les tests sur rongeurs sont les premières étapes de toute mise sur le marché de médicament, de l'évaluation de la toxicité de substance chimique impliquée dans l'agro-alimentaire et de beaucoup de recherche en génétique grace au phénotypage. L'activité motrice est un indice princeps du comportement et donc sa quantification un enjeu important. Je me suis donc intéressé à la mesure 3D du mouvement du rongeur sous diverses conditions, du laboratoire d'anatomie comparée au vol parabolique en apesanteur. Des modèles anatomiques 3D ont été développés et couplés aux méthodes d'estimation de mouvement à partir de la vidéo. Ces recherches expérimentales ont conduit à la mise en place d'une nouvelle plateforme d'analyse construite autour d'un réseau de caméras et de cinéradiograpie biplanaire.

Table des matières

5

Introduction

Il est beaucoup plus fréquent d'entendre parler de capture que de mesure de mouvement. On peut y voir l'intuition que derrière le mot mouvement se conçoit un phénomène plus complexe que la donnée de marqueurs qui en constitue aujourd'hui la norme d'instrumentation rigoureuse. Si le marqueur est quantifiable, le mouvement conserve une qualité supplémentaire à explorer. Ce document retrace ainsi les travaux de recherche que j'ai encadrés ces dernières années sur cette notion de capture de mouvement, à travers les outils scientifiques que sont la mesure et le modèle. Le choix du mot "mesure" renvoie d'abord à une ambition de métrologie. Une partie de la motivation de ma démarche a été de proposer des instruments pour l'étude du mouvement qui soient à la fois techniquement novateurs et si possible fiables au-delà du champ de l'informatique. Le terme de "modèle" représente quant à lui une tentative d'aller au-delà de la mesure technique pour rendre compte d'une compréhension scientifique de la complexité du mouvement. Les thèmes que j'ai abordés dans cette optique recouvrent en particulier des méthodes de suivi vidéo et des systèmes de synthèse graphique. Mes activités ont été initialement dédiées à l'animation 3D, puis se sont progressivement tournées vers des enjeux liés à l'anatomie.

Mesure et Modèle, Informatique et Biologie, Technologie et Science, sont des dialectiques qui ont accompagné ma démarche. Si Héphaïstos est le dieu de la Robotique, celui du mouvement doit être un proche cousin. Dans mes réalisations en informatique, j'ai tenté de traiter autant que possible mesure et modèle comme un seul sujet : le modèle permet soit d'aborder la mesure d'une manière orientée par une approche d'analyse par la synthèse (*model-based analysis*), soit de produire un résultat de synthèse construit à partir d'une utilisation directe de données (*data-driven synthesis*). La double orientation vers l'Informatique et la Biologie m'a conduit à produire des résultats séparés, d'un côté en Animation 3D, et de l'autre, plus expérimental en Anatomie. Les deux domaines ne sont pourtant pas sans lien : ce sont les résultats visuels obtenus en Animation qui ont suscité l'intérêt des Biologistes. Ma pratique du travail scientifique dans ce nouveau contexte m'a montré que le chemin à parcourir reste plus long qu'une simple transposition des outils informatiques. Et même si fournir de nouveaux

outils techniques ouvre des perspectives à l'exploration scientifique, l'exigence de validation a imposé parfois de reformuler les objectifs. Cédant à une certaine appétence humaniste, c'est par une perspective de synthèse pluridisciplinaire que j'envisage à long terme d'aborder la dernière dialectique entre Technologie et Science.

Mesure et modèle de mouvement se retrouvent dans les deux domaines principaux que sont l'Animation 3D et la Biomécanique. Les deux entretiennent des relations multiples, à la fois d'inspiration mutuelle et de mise à distance. Il est naturel que des connaissances sur les structures musculo-squelettiques bénéficient au réalisme d'une animation. A l'inverse, le développement technologique poussé par les enjeux commerciaux de l'image de synthèse apporte des outils nouveaux à l'analyse clinique et la modélisation du mouvement. Et pourtant, les enjeux en Animation et en Biomécanique restent différents. Un modèle est toujours une restriction de la réalité. Le confronter à la mesure permet d'en évaluer la pertinence. On verra dans ce document plusieurs degrés de mesure de la réalité du mouvement articulé, allant d'une simple vidéo documentaire à la prise de vue 3D du squelette en mouvement. Mais sur ce continuum, un modèle est toujours venu interroger la donnée captée pour en délivrer une mesure porteuse de sens.

La notion de modèle pour l'analyse est un concept classique en vision par ordinateur. Il l'est moins en Biomécanique et généralement en Biologie. Par tradition, ces domaines reposent plutôt sur un traitement sériel, partant de la mesure puis opérant un filtrage pour séparer le signal informatif du bruit inhérent à toute mesure. Tout modèle est une prise de décision. Il renforce les contours d'un signal mais génère un légitime scepticisme car il risque de masquer certains aspects du signal non pris en compte dans la modélisation. L'analyse par la synthèse n'est jamais définitive.

Le document s'organise autour de trois thématiques principales : la construction de modèles paramétriques pour l'animation 3D, l'intégration de données réelles en animation physique et les approches expérimentales pour l'étude de l'anatomie en mouvement des petits vertébrés.

CHAPITRE 2

Espaces de mouvements

2.1 PRÉAMBULE

Nos contributions en animation 3D ont d'abord porté sur la recherche d'espaces paramétriques optimaux pour mesurer et générer le mouvement articulé. L'optimalité est à prendre ici au sens de la recherche d'une réduction de dimensions qui préserve au mieux la qualité du mouvement. Cette démarche est inspirée de mes travaux précédents sur les mouvements faciaux en production de la parole (figure 2.1). Les travaux sur les visages parlants avaient été effectués pendant ma thèse au sein de l'Institut de la Communication Parlée (ICP, maintenant intégré au laboratoire GIPSA à Grenoble) et pendant un séjour postdoctoral au Georgia Institute of Technology (Atlanta, GA). Ils ont contribué à montrer qu'en dépit de la complexité de la musculature du visage, quelques gestes prototypiques émergent. Ces gestes, décrits de manière qualitative jusqu'alors en phonétique, ont été numériquement extraits par analyse statistique de données de capteurs 3D placés sur le visage d'un locuteur. L'identification de ces gestes a donné lieu au développement d'un modèle 3D paramétré de visage parlant[RBB00]. Pour l'aspect mesure, ce modèle a aussi été appliqué avec succès au problème technique de suivi automatique des mouvements faciaux à partir de la vidéo, la réduction de paramètres permettant une optimisation plus robuste [RE01].

Pour les visages, la réduction paramétrique a permis non seulement de rendre le problème du contrôle et du suivi plus simples d'un point de vue calculatoire, mais elle a aussi permis d'attribuer un sens aux paramètres, apportant un élément de compréhension du mouvement. Après le visage, la démarche a donc été d'appliquer la même méthodologie au mouvement articulé du corps. L'idée était, là aussi, de trouver si un nombre optimal de degrés de liberté pouvait être dégagé pour le mouvement, et si ces degrés avaient un sens interprétable. La conclusion est qu'une telle transposition n'est pas si directe. Même si la réduction des degrés de liberté a été décrite en neurophysiologie comme une stratégie du cerveau pour la simplification du contrôle moteur

[TM07] [Ber10], la grande variabilité des mouvements articulés du corps rend difficile l'extraction d'un codage aussi ciblé que pour le visage. La recherche d'espaces optimaux s'est donc surtout déclinée autour d'applications pratiques. Parmi elles, deux approches basées sur l'analyse vidéo du mouvement sont présentées ici. Toutes deux visent en particuliers le mouvement animal, une première dans un cadre d'animation 3D et une seconde plus axée sur la mesure anatomique. Dans les deux cas, l'espace de mouvement a pour fonction de modéliser la variabilité du signal perçu pour améliorer la robustesse de l'analyse. Les contributions sur la recherche d'espace de mouvement ont ensuite porté sur des outils de génération à partir de données de marqueurs 3D et non plus de vidéo. Les espaces de mouvements y sont exploités comme paramétrage plus efficace pour la composition procédurale par cinématique directe et inverse.
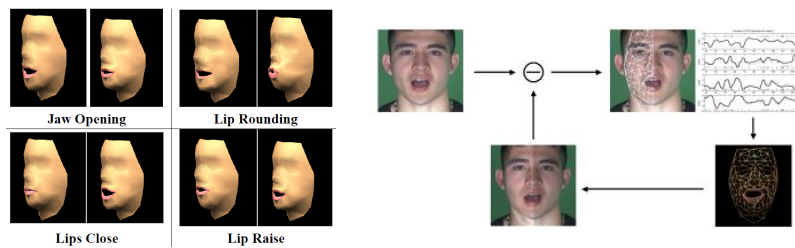
FIGURE 2.1 – *Gauche : Les 4 degrés de liberté des mouvements faciaux de la parole, Droite : Suivi vidéo monoculaire d'un locuteur*

Dans tous les travaux, l'outil de base pour la modélisation est à chaque fois l'analyse statistique multidimensionnelle, en prenant l'Analyse en Composantes Principales (ACP) et la régression multiple comme point de départ. En fonction de la nature propre des données considérées, des adaptations ont été abordées ainsi que des extensions vers des méthodes dites "à noyaux", inspirées de l'apprentissage automatique.

2.2 ESTIMATION DU MOUVEMENT À PARTIR DE LA VIDÉO

La nomenclature habituelle en vision par ordinateur sur le suivi de mouvement distingue deux approches : les méthodes discriminantes et les méthodes génératives [MHKE11]. Les méthodes discriminantes s'apparentent à la régression multiple. Une phase d'apprentissage met en rapport un jeu de données observées (ici, la vidéo) avec un jeu de données à prédire (ici, les paramètres d'articulations). Le modèle déduit sert ensuite à prédire automatiquement les paramètres visés à partir de nouvelles données d'observation. Les méthodes génératives quant à elles s'appuient sur un modèle de synthèse contrôlé par des paramètres d'état qui peuvent être les paramètres articulaires ou d'autres qui leur sont liés. On va alors chercher à aligner par optimisation le modèle

sur le signal observé. Dans le cas du mouvement articulé, on va donc être amené à construire un modèle 3D de l'objet suivi dont la projection en image 2D sera alignée sur l'observation vidéo.

Les sections suivantes décrivent deux approches, respectivement selon un schéma discriminatif et génératif. Ces deux approches d'estimation de mouvement à partir de la vidéo ont été développées en particulier pour le mouvement animal avec l'argument principal de s'affranchir de la mesure par marqueurs, peu adaptée à leurs cas. La première méthode, discriminante, a cherché à prédire le mouvement articulé à partir d'une vidéo unique. Elle s'appliquait à un cadre d'animation 3D où au final, seule la vraisemblance visuelle importe. Ces résultats ont sollicité une demande d'application pour la mesure de mouvement des rongeurs. L'expérience a montré qu'elle ne répondait pas aux exigences de mesure, ni vis à vis de la variabilité du signal d'entrée, ni pour la qualité de l'information articulatoire attendue. Après un développement autant expérimental que technologique, nous nous sommes donc tournés vers une méthode générative pour laquelle un modèle précis de l'objet suivi a été développé, en s'appuyant sur une donnée d'entrée plus riche (maillages 3D en mouvement). Cette source de donnée satisfait l'exigence de l'absence de marqueurs mais introduit des difficultés métrologiques auxquelles nous avons proposé de répondre par une nouvelle approche de modélisation.

2.2.1 Une méthode discriminante à partir d'une vue 2D unique

A côté de son incarnation morphologique, la vraisemblance d'un mouvement passe aussi par sa rythmicité, son "timing". C'est cette dimension qui a été explorée ici à travers une application de génération d'animation à partir de la vidéo. Peut-on mesurer le rythme du mouvement dans une vidéo et le transposer à une séquence d'animation 3D ? La motivation initiale repose aussi sur un constat pratique : il est difficile d'instrumenter un animal sauvage avec des marqueurs alors qu'un nombre important de documents vidéo existent. Une transposition simple de l'approche marqueurs à la vidéo consisterait à appliquer une poursuite d'indices visuels dans la séquence d'images. Sans hypothèse supplémentaire, cela ne permet pas de remonter à une information 3D, et surtout la qualité de la plupart des documentaires vidéo standards ne permet pas d'extraire des trajectoires sur de longues périodes (occultations, variation d'éclairage). L'idée a donc été de mettre en place une régression statistique globale des poses articulaires 3D directement, où chaque image de la vidéo est prise comme un vecteur d'observation dont les pixels sont les paramètres (figure 2.2).

Au coeur de tout problème de regression multiple se trouve l'inverse d'une matrice de covariance. Pour notre cas, cette matrice réalise un modèle image des variations dues au mouvement. Les images présentent naturellement une grande redondance spatio-temporelle. Cela a conduit à considérer une régularisation de l'analyse. En se focalisant, sur la locomotion, et en filtrant les données vidéos par ACP, les nomogrammes ont montré que les seules composantes utiles pour la prédiction de mouvement étaient celles portant un sens sur le mouvement 2.3. Le principe de régression amène à demander à l'utilisateur de fournir un exemplaire de pose articulaire 3D pour chaque individu
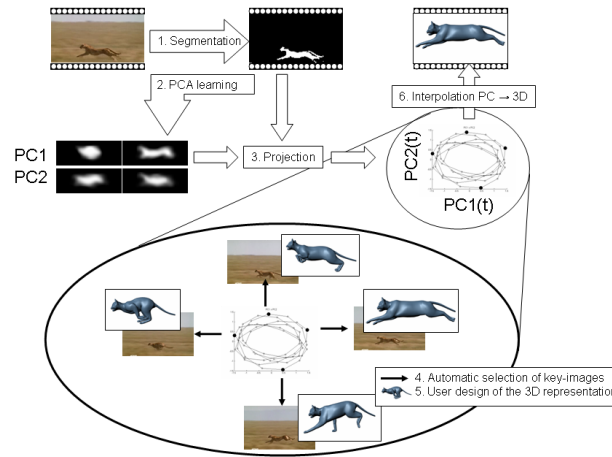
FIGURE 2.2 – *Méthode discriminante de prédiction de mouvement articulaire [FRDC04].*

image sélectionné pour l'apprentissage du modèle. Bien sûr, toutes les trames de la vidéo ne sont pas utilisées pour l'apprentissage. En se basant sur des critères numériques de conditionnement de matrice, la formulation de la régression par Fonction à Base Radiale (RBF) a permis de dégager un principe automatique de sélection optimale des individus à identifier parmi les trames. Cette démarche répond à un critère pratique pour limiter le nombre de pose 3D que l'utilisateur doit fournir pour l'apprentissage.
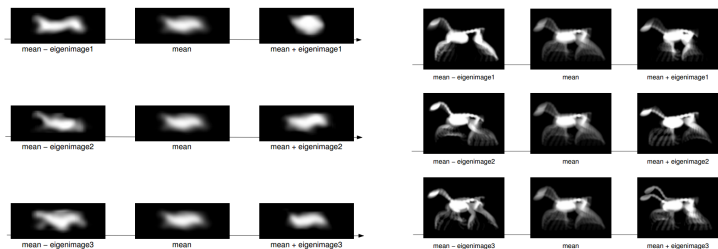
FIGURE 2.3 – *Nomogrammes images dont la source est, à gauche, une vidéo terrain d'un guépard, à droite, une séquence de rendu d'une synthèse 3D*

Cette méthode a permis de générer des animations de marche et de course pour plusieurs quadrupèdes. Le résultat théorique est que, indépendamment même de la prédiction articulaire, le rythme du mouvement est directement mesuré par l'ACP du flux vidéo, fournissant ainsi une métrique originale qui reste cependant difficile à valider d'un point de vue métrologique.

2.2.2 Une méthode générative à partir de données 4D

Le travail précédent permettait d'avoir une estimation globale du mouvement qui s'appliquait bien à un but d'animation 3D, c'est-à-dire fournir un résultat visuel acceptable pour l'œil du spectateur. Ce n'est cependant pas suffisant pour une métrologie biologique des structures squelettiques. La difficulté du dialogue initial avec les biologistes a été que la qualité graphique du résultat d'animation confère une forte impression visuelle de vraisemblance qui masque les faiblesses de la maîtrise du sens physique de la mesure. Il a donc fallu passer par une phase de remise en question des résultats précédents pour reconstruire une approche adaptée. Ce travail a été l'objet du projet ANR Kameleon que j'ai coordonné, avec le Muséum National d'Histoire Naturelle, la laboratoire de Biomécanique de l'Université de Bretagne et le laboratoire de Neurosciences LNRS à l'Université Descartes. Les aspects expérimentaux et anatomiques de ce projet sont décrits plus loin au chapitre 4. Est décrite ici la partie du projet qui a conduit a proposé une nouvelle méthode générale en suivi vidéo. On précise juste que le but ultime est de fournir à partir de vidéo une prédiction du mouvement squelettique 3D. La contrainte sans marqueur s'applique encore mais il est évident qu'une seule vue ne permet pas d'aborder une information 3D de manière automatique. Le choix s'est donc porté sur les systèmes calibrés multivues, en particulier sur les nouvelles techniques délivrant un flux de maillages 3D par reconstruction via enveloppe visuelle convexe [Lau94]. Ces méthodes présentent l'avantage de délivrer une résolution spatio-temporelle 3D qui est limitée par la seule performance des caméras et donc améliorable avec les caractéristiques techniques de ces dernière. En l'occurence, il est courant maintenant d'avoir des résolutions de l'ordre du megapixels pour des fréquence de l'ordre de 100 Hertz, ce qui convient au mouvement animal même aussi véloce que celui des rongeurs.
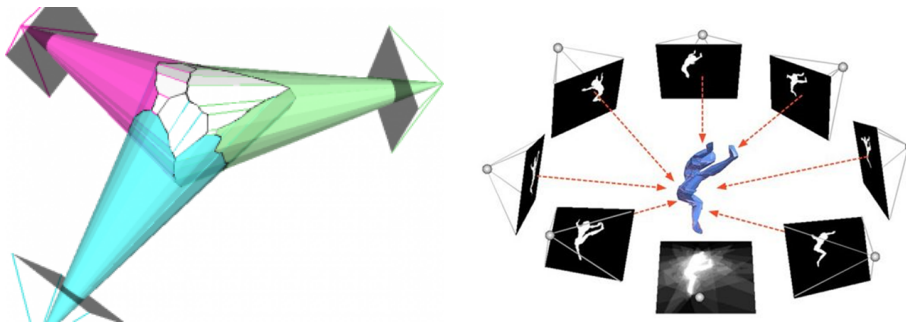
FIGURE 2.4 – *Reconstruction 3D par enveloppe visuelle. Les silhouettes calibrées génèrent des cônes 3D dont l'intersection fournit une reconstruction de l'objet filmé (d'après [AFM⁺06]).*

La caractéristique des méthodes de reconstruction par enveloppe visuelle est de délivrer une reconstruction pour chaque trame des vidéos synchronisées. En ce sens, il n'y a pas de cohérence temporelle d'un maillage à l'autre. Il est donc impératif de réaliser un suivi de ces maillages pour imposer une sémantique anatomique à l'analyse.

Des méthodes automatiques existent à base de suivi d'éléments de surface [CBI10]. Elles présentent l'avantage de s'adapter à toute topologie mais, en restant locale, sont sensibles à un fort bruit de mesure potentiel, surtout en expérimentation animale. Une approche plus globale consiste à utiliser un modèle complet de référence du sujet suivi que l'on met en correspondance avec le signal 3D du flux de maillage non temporellement cohérent. Le type de modèle le plus simple se construit autour d'un squelette d'animation et décrit les déformations d'une surface 3D par un algorithme standard de "skinning" linéaire [GSdA⁺09]. Un tel modèle a été difficile à développer pour le rongeur. Ces animaux présentent des déformations importantes et contrairement aux formes humaines, n'ont pas de membres bien marqués (figure 2.5).
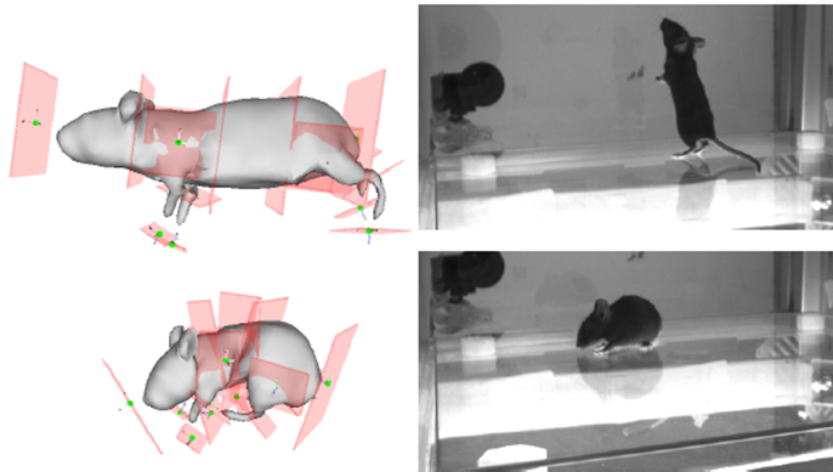
FIGURE 2.5 – *Variabilité des déformations chez la souris*

.

Nous nous sommes donc tournés vers une autre représentation des déformations de la surface 3D, plus souple que le "skinning" linéaire, et ne nécessitant ni la définition d'un squelette de contrôle, ni un réglage fastidieux des poids d'interpolation. Nous avons développé une représentation morphologique à base de maillage de contrôle (ou "cage") et déformation barycentrique. Le principe général consiste à déduire les déformations d'un maillage de référence à haute densité, le patron (*template*), des déformations d'un maillage de contrôle à plus faible densité, la cage. Le principe de suivi consiste alors à ajuster pour chaque trame les paramètres de la cage pour que le maillage patron s'aligne sur le maillage issu de la reconstruction 3D multivues. Se faisant, la cage et le maillage de référence implémentent une cohérence temporelle. Pour le calcul des poids d'interpolation, nous avons choisi l'algorithme des coordonnées de Green [LLCO08]. La méthode de suivi délivre donc un flux de maillage à haute définition avec cohérence temporelle. Des points de mesure sur le maillage de référence permettent de créer des marqueurs virtuels qui peuvent servir à produire le mouvement d'un squelette articulé par cinématique inverse.

La difficulté est de trouver une représentation de la cage qui soit suffisamment détaillée pour couvrir la variabilité des déformations (une simple boîte englobante avec 8 sommets ne conviendrait pas) mais trop de paramètres entraînent une instabilité pour le processus d'optimisation du suivi (problème de "many-to-one"). Nous avons donc opté pour une représentation d'inspiration anatomique, s'apparentant à des plans articulaires (figure 2.5). Une procédure automatique construit automatiquement le maillage de la cage à partir des plans. Le paramétrage est donc réduit à l'ensemble des positions rigides de ces plans. Nous avons introduits le terme d'*Oriented Quad Rigging* (OQR) pour désigner ce paramétrage des formes articulées à partir d'un ensemble de plans orientés dans l'espace.

Un tel paramétrage présentait encore trop de degrés de liberté. Afin de réduire l'espace de recherche du processus d'optimisation, un ensemble de poses clés a servi à un apprentissage par Modèle de Processus Gaussiens à Variables Latentes (*Gaussian Process Latent Variables Models*, GPLVM [Law05]). Ainsi, les variables d'états de l'optimisation ne sont plus directement les paramètres géométriques des OQR, mais des variables de haut niveau, s'apparentant à des composantes principales. L'ensemble du processus de suivi vidéo est résumé sur la figure 2.6. Pour le rongeur, ces poses clés ont été éditées par l'utilisateur. Un ensemble d'une dizaine de poses a permis d'obtenir des résultats préliminaires. Ils ont cependant montré qu'il était nécessaire d'envisager un nombre plus important de poses d'apprentissage. Ce point est illustré par la suite à travers une évaluation sur le mouvement humain.
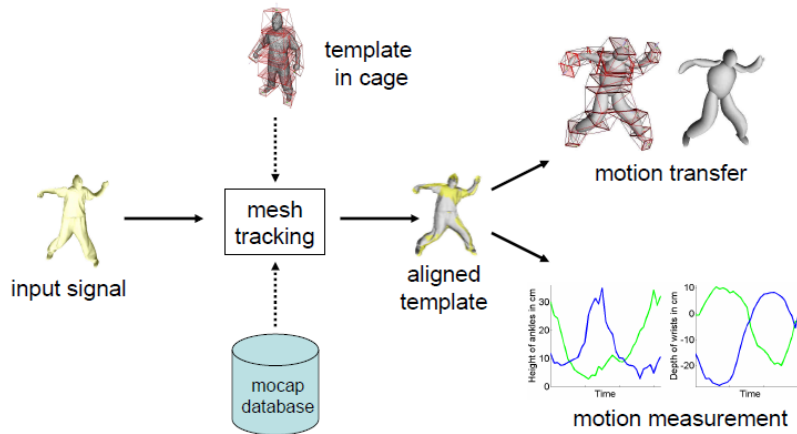
FIGURE 2.6 – *Suivi vidéo par une méthode générative [DCRB12].*

Pour le mouvement humain, une base de donnée vidéo multivues a été proposée par [SBB10]. Cette base HUMAN-EVA présente l'avantage de fournir à la fois des vidéos calibrées d'un sujet humain marchant devant les caméras, et des données synchronisées issues d'un système de capture de mouvement à base de marqueurs 3D. Il est donc possible de fournir une évaluation chiffrée d'un algorithme de suivi de mouvement tel que le nôtre. Contrairement à l'étiquetage manuel fait sur les rongeurs, un apprentissage automatique a pu être développé. La description des primitives OQR s'interprète

facilement par rapport à un squelette d'animation. Nous avons donc générer de longue séquence *a priori* de trajectoires de ces OQR en définissant un couplage rigide avec un squelette d'animation et des données issues de la base mise à disposition par le CMU [CMU02]. La qualité d'image de la base vidéo HUMAN-EVA ne permet pas d'obtenir de silhouettes très nettes, et de ce fait les maillages 3D reconstruits restent très bruités. Malgré cela, nos résultats (figure 2.7) ont montré que notre système atteignait des résultats atteignant l'état de l'art pour l'analyse de mouvement sur la base HUMAN-EVA [GSdA$^+$09] [CGMA10], et ce sans utiliser d'information colorimétrique, ce qui souligne la robustesse de l'approche choisie.
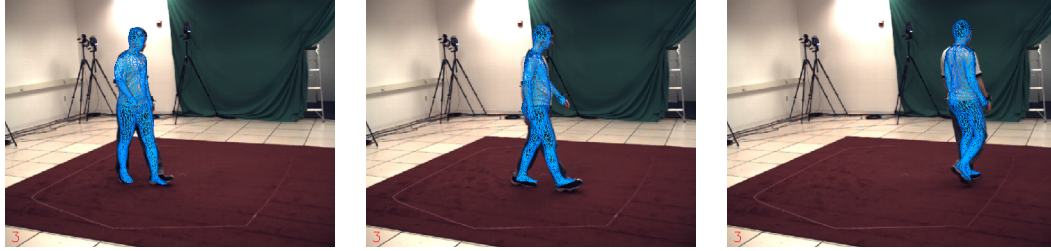
FIGURE 2.7 – *Résultats du suivi vidéo par modèle génératif sur la base HUMAN-EVA [DCRB12].*

2.3 MODÈLE DE GÉNÉRATION DE MOUVEMENT

Les modèles précédents s'inscrivait dans l'optique de capter le mouvement à partir de la vidéo. Ils présentaient en ce sens un intérêt pratique de mesure et génération 3D de mouvement sans marqueur. Nous nous intéressons ici à des approches pour étudier la variabilité du mouvement humain de manière plus précise et utilisons des données classiques à base de marqueurs. Nos travaux utilisent la base de données de près de 2605 séquences que propose depuis 2002 Carnegie Mellon University[CMU02]. Nous avons utilisé cette base pour développer deux types de modèles de mouvement, l'un pour la compression de données de capture et l'autre pour l'édition de pose articulée. Une autre piste a été explorée avec l'analyse modale des structures articulées, permettant de proposer un modèle de génération de mouvement à partir de l'analyse de données non plus cinématiques mais mécaniques.

2.3.1 Analyse en Géodésiques Principales des données articulaires

La motivation initiale de ce travail était de caractériser les degrés de liberté du mouvement d'un personnage articulé. Il s'est concrétisé à travers une application de compression de données d'animation de personnage qui repose sur une méthode de génération de mouvement apprises sur les données. Les données de départ sont ici les trajectoires articulaires classiques issues des systèmes à base de marqueurs : position

et rotation globale d'une articulation de référence et rotations locales d'une hiérarchie d'articulations. Chaque séquence de mouvement de la base CMU est paramétrée par un squelette classique avec 90 degrés de liberté (3 pour chacune des 30 articulations). D'un point de vue statistique, chaque trame d'une séquence de mouvement est une observation et chaque paramètre de rotation une variable. Par souci d'homogénéité numérique des variations, la position et l'orientation de l'articulation de référence ne sont pas intégrées à l'analyse.

Le paramétrage initial issu de la base de données suit une description en angles d'Euler pour les rotations aux articulations. Pour des mouvements d'amplitude modérée tels que la marche, des résultats d'application de PCA ont été obtenus par [GBT04] pour obtenir des modèles de génération de mouvement. Cependant, pour une collection de mouvements présentant un éventail plus élargi de variations, appliquer directement une analyse statistique classique soulève des problèmes. Typiquement, les angles d'Euler peuvent présenter des variations numériques qui sont dues à leurs singularités mathématiques propres et non au mouvement (périodicité, non-unicité dans l'application de $\mathbb{R}^3$ vers $SO(3)$ et "gimbal lock"). Les quaternions unitaires ou les cartes d'exponentielles (*exponential maps*) fournissent un paramétrage plus adapté pour les rotations dans l'espace.
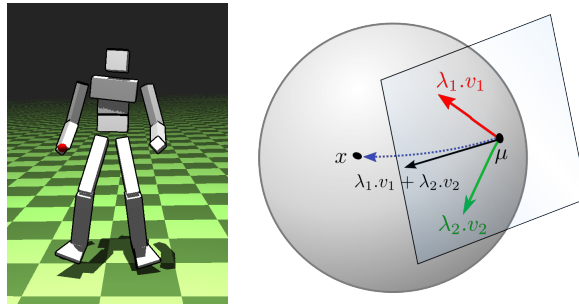
FIGURE 2.8 – *Les degrés de liberté du modèle articulé, un point dans $SO(3)^n$ (gauche). Une illustration de l'approximation des géodésiques dans le plan tangent de $SO(3)$ (droite)*

L'ACP nécessite une métrique quadratique bilinéaire liée au produit scalaire d'un espace Euclidien afin de quantifier les variations numériques des paramètres. On peut appliquer l'ACP telle quelle à des angles d'Euler mais avec les problèmes de singularité évoqués. Les quaternions unitaires ne s'adaptent pas bien non plus à un cadre d'ACP car l'aspect projectif linéaire de l'ACP ne permet pas de maintenir l'unicité de norme requise par les quaternions. A la suite des travaux de [FLJ03] et [Pen06], les solutions pour appliquer une statistique sur des éléments de rotations sont connues. La méthode consiste à tirer bénéfice de la structuration de $SO(3)$ en groupe de Lie. De ce groupe de Lie, on dérive la possibilité de calculer un paramétrage homéomorphe à $\mathbb{R}^3$ par logarithme pour passer à l'espace tangent $\mathfrak{so}(3)$ des matrices de rotation qui s'apparente au vecteur de vitesse angulaire instantanée. Cet espace comprend toutes les directions initiales, vecteurs directeurs des géodésiques, à l'endroit de la moyenne

dite intrinsèque [FLJ03]. L'extension de la PCA consiste alors à trouver la suite de géodésiques qui maximise la variance des projections des observations sur celles-ci, conduisant à l'Analyse en Géodésiques Principales (AGP). Cette étape de projection n'a pas de solution exacte simple car on est toujours sans produit scalaire. L'approximation généralement faite, et qui s'apparente à une linéarisation de l'exponentielle, consiste à réaliser une ACP dans l'espace tangent (figure 2.8). Pour la reconstruction du signal, on revient ensuite à des rotations de $SO(3)$ en utilisant simplement l'exponentielle de matrice, aboutissant à la formule dite de Rodrigues pour ce cas précis des rotations. Nous avons suivi ce cadre et montré qu'il était applicable au cas $SO(3)^n$ qui caractérise les données de l'ensemble des articulation d'un squelette d'animation.

Pour des mouvements simples de locomotion, on retrouve des paramètres porteurs de sens tel que l'oscillation des bras ou des jambes. Cependant, il reste difficile d'apporter une sémantique universelle sur la gestuelle corporelle. De ce travail, nous avons quand même retenu un premier résultat qui peut être qualifié de théorique et qui est résumé sur les figures 2.9. Elles montrent des histogrammes de la répartition des séquences en fonction du nombre de composantes nécessaires pour atteindre au moins 95% de la variance de la séquence. Nous avons distingués deux cas : toutes les séquences de la base et les séquences liées à la marche seulement. Les pics correspondent donc aux cas les plus fréquents de nombre de composantes nécessaire pour atteindre 95% de la variance et donc qu'une dizaine de composante linéaire sont suffisante pour toute marche, une vingtaine pour toute séquence. Par rapport aux 90 degrés de liberté initiaux, ce chiffre de 20 reste une majoration intéressante, même en prenant en compte des contraintes biomécaniques telles que le fait que genoux et codes n'ont qu'un seul degré de liberté naturel aboutissant à un chiffrage initial inférieur à 90.
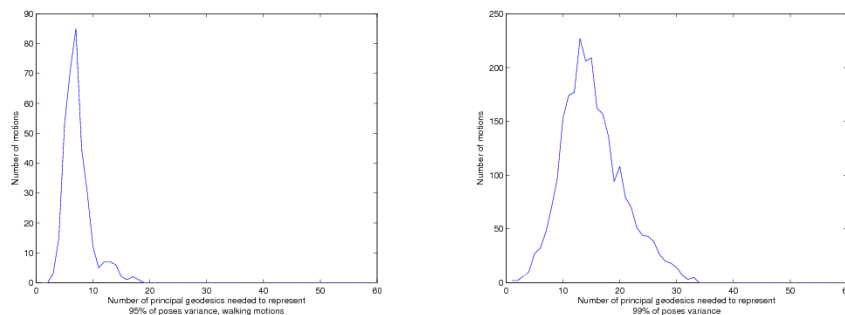
FIGURE 2.9 – *Distribution d nombre de composantes pour atteindre 95% variance sur la base CMU pour la compression de données, marche (gauche) et toutes mes séquences (droite) [TWC⁺09]*
.

Identifier une borne supérieure de la complexité des degrés de libertés a trouvé une application pour la compression de mouvement. C'est dans ce cadre que le travail a été publié. Le principe est d'utiliser le paramétrage en géodésiques principale comme espace de recherche pour une tâche de cinématique inverse. Pour une séquence don-

née, les quantités encodées restent ainsi la trajectoire de l'articulation de référence, les trajectoires des effecteurs finaux (mains et pieds) et l'espace des vecteurs propres de l'AGP. Le décodage consiste dans la relecture des trajectoires et l'application de la cinématique inverse dans l'espace AGP. L'expérience a montré qu'un très fort taux de compression pouvait être atteint, tout en préservant une bonne qualité de reconstruction de mouvement. Le choix d'avoir recours à un schéma de cinématique inverse et de synthèse de mouvement permet de garantir une meilleure qualité visuelle en évitant le phénomène de glissement aux contacts.

2.3.2 Edition de mouvement par Modèle GPLVM

Le travail précédent exploite les modèles de mouvement type AGP via un principe de synthèse par cinématique inverse. Chaque base de vecteurs propres est optimisée pour la séquence de mouvement qu'elle encode. Cette spécialisation satisfait bien un but de compression mais ne se généralise pas bien pour l'idée d'édition de mouvement que sous-tend la méthode. Nous nous sommes donc intéressés à mettre en œuvre une construction de modèle qui prenne en compte la base de donnée la plus large possible afin de fournir un outil d'édition de mouvement le plus générique possible et offrant des mouvements "naturels", au sens appris de données réelles.

Le travail de [GMHP04] avait introduit une telle approche probabiliste de la cinématique inverse. Il repose sur l'utilisation de la méthode de régression multidimensionnelle dite à "noyaux" des modèles GPLVM déjà vus vi-dessus. Pour les GPLVM, on retrouve une recherche de réduction paramétrique type ACP (variables latentes) pour laquelle on remplace la projection sur les axes principaux par des distances (processus gaussien) dans l'espace des composantes aux individus des données d'apprentissage. Pour un ensemble d'apprentissage de N individus, le calcul du modèle qui s'apparente donc à une inversion de matrice est en $O(N^3)$. Ce coût devient prohibitif et ne permet pas d'envisager d'inclure toutes les trames de la base CMU utilisée.
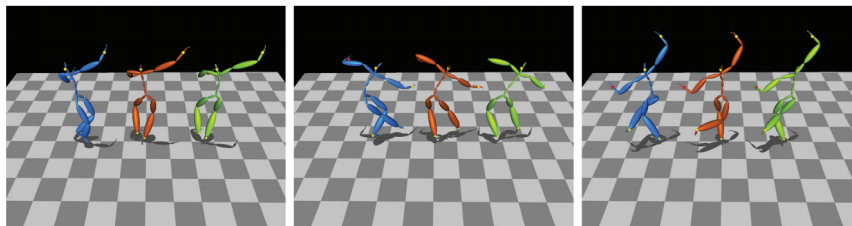
FIGURE 2.10 – *Edition de mouvement par Modèle GPLVM. Comparaison pour trois examples de poses, entre la méthode [GMHP04] (bleu, gauche), la pose originale (orange, milieu), et notre méthode (vert, droit) [WTR11].*

Nous nous sommes donc intéressés à une approche permettant d'intégrer progressivement des données d'apprentissage au modèle GPLVM par une phase de classification préalable. Toutes les trames ne sont pas intégrées mais seulement les représentants de

classe. De plus nous avons raffiné l'introduction progressive de données en nous basant sur un procédé d'apprentissage à partir d'un ensemble d'amorçage (*active set*). Le mécanisme d'intégration s'appuie sur le principe de complément de Schurr nécessaire pour remettre à jour par bloc l'inversion de la matrice de covariance. Ce travail a montré que toute une base de 1265 séquences correspondant à plus de 2 millions de trames pouvait être traitée hors-ligne pour obtenir un modèle temps-réel d'édition de mouvement. Le principe d'édition de mouvement repose toujours sur un schéma de cinématique inverse. Les résultats obtenus montrent une capacité de reconstruction de pose à partir de la donnée des effecteurs finaux plus vraisemblable que la méthode proposée par [GMHP04]. L'explication repose sur la plus grande couverture du jeu d'apprentissage pour la construction du modèle à variables latentes.

2.3.3 Génération de locomotion par analyse modale

Nous avons exploré ici l'hypothèse d'une organisation du mouvement selon la combinaison de modes principaux mais à travers une autre méthode : l'analyse vibratoire, ou analyse modale, des structures articulées. L'analyse modale consiste à décomposer un système mécanique en un ensemble d'oscillateurs découplés, alignés sur les fréquences de résonnances du système. Le système mécanique considéré correspond ici à une hiérarchie de solides articulés, liés entre eux par des articulations présentant une raideur élastique, principalement en rotation, la raideur en translation étant moins vraisemblable physiologiquement. En pratique, seule l'inverse de la raideur intervient (*compliance*), cet inverse étant donc nulle pour les degrés de liberté en translation. Les modes sont alors calculés par décomposition en valeurs singulières de la matrice inverse de raideur, pondérée par l'inverse de la matrice inertielle des solides. Une transformation est appliquée pour transporter les forces de raideur des ressorts angulaires aux articulations jusqu'aux repères des solides. Une restriction sur les degrés de liberté est aussi introduite par projection pour conserver l'expression de contraintes cinématiques.

Le cadre s'éloigne de l'analyse de données mesurées car les modes sont ici calculés à partir de la modélisation directe du système articulé. Les modes reflètent donc avant tout les choix de modélisation. Les éléments rentrant en jeu sont : les matrices inertielles, les degrés de liberté aux articulations, les raideurs des ressorts angulaires. Pour les premières, un modèle géométrique et une hypothèse de répartition uniforme de la densité délivrent des valeurs plausibles. Le choix des degrés de liberté a découlé d'une analyse anatomique simple, en attribuant des liaisons pivots aux articulations des rigides situés sur coudes et genoux, et rotule partout ailleurs. La décision sur les raideurs était la plus difficile à tirer d'une quelconque observation : en l'absence d'étude de données expérimentales, elles ont toutes été définie uniformes.

Au final, les modes obtenus sont recombinés pour créer une animation. Des résultats intéressant ont été obtenus suggérant que les modes issus de l'analyse modale sont représentatifs de ce que l'on peut attendre de la locomotion. Ainsi, la manière la plus naturelle de combiner ces modes consiste à les coupler avec un générateur sinusoïdal, un réglage des phases et amplitudes permet d'implémenter des animations illustrant

différents styles, et plus particulièrement allures, pour les quadrupèdes. Ce résultat liant l'interprétation des modes à la locomotion tend à confirmer que la locomotion est le mode d'expression d'une certaine optimalité dans le mouvement. La question sous-jacente reprend un débat intéressant entre forme et fonction, à savoir si la forme crée la fonction ou si la fonction recherchée modèle la forme.
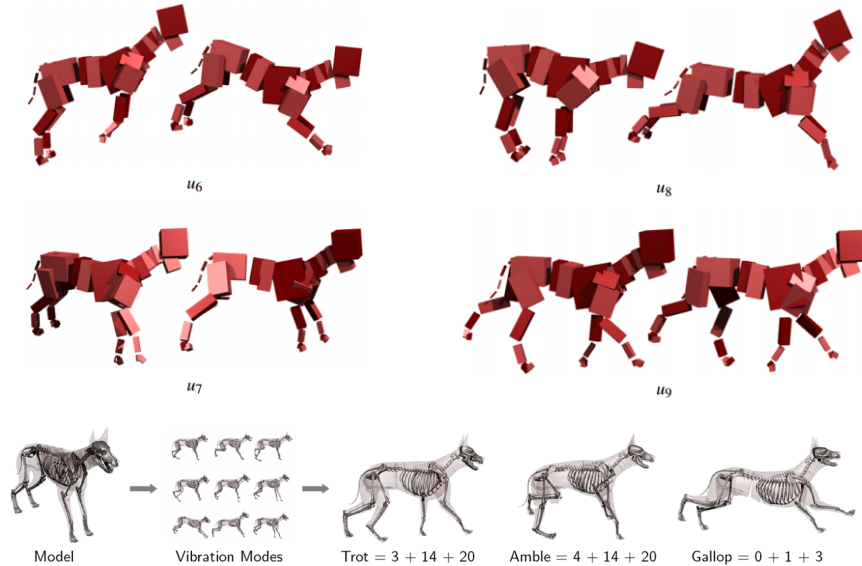
FIGURE 2.11 – *Modes vibratoires obtenus par Analyse Modale (haut), composition procédurale de mouvement (bas)[KRFC09].*

A noter que parallèlement au travail sur les personnages articulé, un travail sur les arbres a permis de montrer que l'analyse modale fournis aussi un bon cadre pour animer le mouvement des arbres sous le vent. En plus d'une méthode procédurale de composition de mouvement comme pour les personnages articulés, un pré-calcul de la projection sur les modes d'une charge directionnelle d'une impulsion de vent a permis de fournir un système intuitif d'animation [DRBR09].

CHAPITRE 3

Données réelles en animation physique

3.1 PRÉAMBULE

Bien que s'appuyant sur un modèle mécanique, le travail sur l'analyse modale fournissait une contribution d'abord cinématique. Les sections suivantes décrivent des approches inscrites plus directement en animation physique. Mes contributions à ce vaste domaine ont continué à suivre une démarche de couplage entre mesure et modèle, en s'intéressant en particuliers à l'intégration de données réelles à la simulation.

Depuis les travaux pionniers de Witkin et Kass [WK88] puis de Raibert et Hodgins [RH91], la simulation physique a suivi deux orientations : l'optimisation par contraintes spatio-temporelles (*space-time constraints*) et le contrôle par suivi de trajectoire (*Proportional-Derivative control*, ou *PD control*). Les développements récents en animation physique de personnages mettent régulièrement en avant comme argument positif le fait de ne pas dépendre de données de mouvement [ABdLH13], [MWTK13]. Prométhée ou Frankenstein, cette idée procède sans doute du vertige de créer la vie *ex nihilo*. Il reste surprenant d'un point de vue scientifique de voir la donnée réelle quasiment vue comme une faiblesse. Il est clair que la simulation physique, en tant qu'implémentation des lois classiques de la mécanique, peut s'affranchir de données *a priori*. C'est même là l'essence universelle de ses lois. La difficulté qui reste est la mise en œuvre du contrôle, savoir comment orchestrer dans le temps l'application des couples actifs aux articulations et intégrer les forces de contacts. Les méthodes d'animation physique déroulent alors toute une série de règles et contraintes que doit suivre une optimisation pour faire émerger automatiquement le mouvement. Les règles proposées dans les travaux récents évoqués ci-dessus relèvent du bon sens et sont légitimes. On peut se dire en définitive qu'elles ont été inspirées de ce qu'on a pu observer des données. Aussi, autant partir directement de ces données. Les trois approches présentées ici explorent cette idée.

3.2 OPTIMISATION D'UN MODÈLE PHYSIQUE DE QUADRUPÈDE

Ce travail constitue ma première incursion dans l'animation physique. Il a été fait en collaboration avec des collègues de l'Université British Columbia. Il concerne le développement d'un simulateur physique de quadrupèdes. Il étend l'approche SIMBICON décrite dans [YLvdP07] et qui portait sur la simulation de marche bipède. L'extension aux quadrupèdes s'est faite en combinant deux modèles bipèdes. Le système de contrôle suit les mêmes règles que SIMBICON : poursuite d'une succession de poses clé par *PD-controller*, combiné à un contrôleur de maintien d'équilibre. Une des clefs du contrôle réside donc dans le choix adéquat des cibles cinématiques, le timing d'enchaînement de ces cibles et la valeur des gains des *PD-controller*. Pour le bipède de SIMBICON, des données issues de capture de mouvement avaient été utilisées. Même en ciblant un animal aussi coopératif qu'un chien, il est difficile de trouver des données 3D complètes fiables *a priori*, en particuliers pour des allures extrêmes comme le gallop.

Pour ce modèle physique de quadrupède, nous avons utilisées des données vidéo collectées par le MNHN sur un paramétrage des allures de quadrupèdes [AHH+07]. Ces données ont été réalisées en extérieur sur une piste de 10 mètres de long avec des chiens de dressage d'une unité cynophile de l'armée de terre. Ce type de protocole a permis d'aborder des allures à grandes vitesse (galop rotatoire et galop transverse) sur une dizaine de cycles de locomotion, allures difficiles à appréhender sur un tapis roulant. La vidéo a été calibrée avec précision afin de fournir des informations métriques et indépendantes du mouvement de caméras. La trajectoire des marqueurs a été extraite en combinant la méthode globale décrite en 2.2.1 avec un suivi local des marqueurs qui avaient pu être posés sur les chiens. Une seule caméra ayant été utilisée, les données marqueurs mesurées étaient astreintes à un plan parallèle au plan sagittal de l'animal. Des informations de profondeur constante ont été attribuées sur la base d'observations anatomiques (figure 3.1).
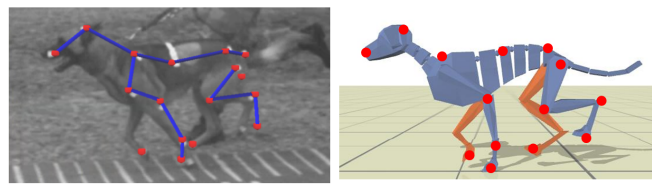
FIGURE 3.1 – *Optimisation d'un modèle physique à partir de données réelles [CKJ+11].*

Une fois ces informations de marqueurs obtenues, elles ont servi à fournir une métrique entre mesures des marqueurs et leur position équivalente sur le modèle physique. Par rapport à cette métrique, les différents paramètres de la simulation ont été optimisés. Les paramètres d'une telle optimisation sont très nombreux, entraînant un risque de minimum local non satisfaisant. Plusieurs méthodes récentes en animation physique

ont montré qu'une optimisation par ordonnancement de priorités peut apporter une solution à ce problème [LMH10]. Dans ce cas, l'optimisation porte sur des paramètres de plus haut niveau que des données de mouvement, dans l'optique de s'affranchir de données a priori. Dans notre cas, il s'est avéré qu'une optimisation simple du problème a convergé vers une solution satisfaisante. Une explication possible peut être avancée par le fait que les données de mouvement apportent une plus grande information que les approches réalisant une optimisation sur quelques paramètres de plus haut niveau. Les figures donnent le détail des comparaisons entre les données réelles et les résultats de simulation (figure 3.2).
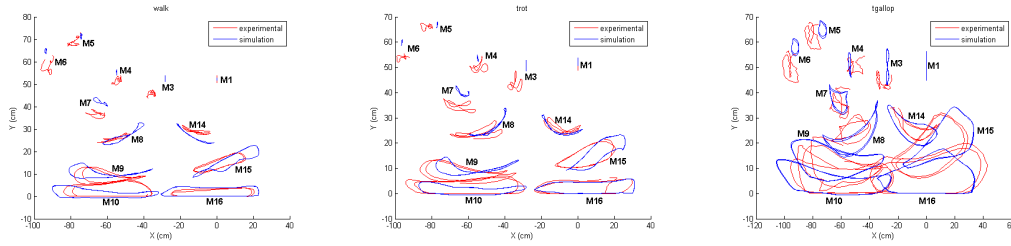
FIGURE 3.2 – *Comparaison entre les données réelles et la trajectoire de marqueurs sur le modèle simulé (marche, trot et gallop transverse)*

3.3   MÉCANIQUE LAGRANGIENNE DES GÉODÉSIQUES PRINCIPALES

Le projet précédent a permis de participer à un premier travail d'animation physique, où l'intégration de données réelles s'est faite de manière externe par rapport au modèle. En prolongeant le travail sur la PGA dans un cadre dynamique, nous avons alors cherché à intégrer plus directement dans la simulation physique les modèles de mouvement issus de l'analyse de données. La PGA et le travail sur le modèle par GPLVM sur base de données étendue avaient montré que les composantes principales (ou variables latentes) pouvait être utilisées comme des paramètres cinématiques de haut niveau. L'intuition est que ces paramètres de haut niveau factorisent certaines synergies dans la coordination motrice. Ainsi, plutôt que de développer des contrôleurs complexes, l'idée explorée ici a été d'utiliser directement ces modes cinématiques de haut-niveau comme variables d'états d'une simulation physique.

La modélisation physique des solides s'exprime selon deux formalismes principaux : en coordonnées maximales ou en coordonnées réduites. Les premières correspondent à un système classique où l'on paramètre chaque solide indépendamment des autres par sa position et son orientation dans un repère inertiel où s'applique les lois de la mécanique classique. Les liaisons d'articulation s'expriment alors par des contraintes type multiplicateurs de Lagrange. Dans le cas des coordonnées réduites, les solides sont paramétrés par un ensemble de variables reflétant exactement les degrés

de liberté d'un système articulé, intégrant directement les contraintes. L'expression des équations du mouvement est modifiée et rentre alors dans le cadre de la mécanique dite Lagrangienne. Le plus souvent c'est cette approche qui est retenue pour l'animation de personnage, en prenant comme paramètre les angles aux articulations. La dérivation des équations de mouvement est alors plus compliquée, nécessitant d'introduire typiquement des termes de Coriolis pour prendre en compte le fait que les variables ne correspondent plus à un repère inertiel.

Avec le projet d'utiliser les coefficients de projection sur géodésiques principales comme variable d'état du système articulé, nous avons donc opté naturellement pour la formulation en coordonnées réduites. La mécanique Lagrangienne implique la dérivation des Jacobiennes nécessaires pour passer des vélocités réduites aux vélocités classiques. Dans notre cas, il y a deux niveaux de dérivation : le passage des coordonnées articulaires aux coordonnées maximales, et le passage des coordonnées géodésiques aux coordonnées articulaires. Le premier passage est classique et se retrouve dans tout texte classique de robotique [MLS94]. Le second est plus spécifique à notre cas mais ne présente pas de difficultés du fait de l'aspect linéaire de la composition des coordonnées géodésiques dans l'espace tangent approprié. Par un choix d'intégrateur explicite, l'équation de mouvement résulte en une simple équation linéaire. Les contacts sont gérés à travers des contraintes unilatérales, intégrées par LCP en reformulant l'équation linéaire de mouvement comme le minimum d'un problème quadratique (condition de Karush-Kuhn-Tucker, abrégée KKT). Nous avons opté pour un choix d'intégrateurs variationnels, introduits en animation par [KCD09] pour la simulation de rigide simple. Une de nos contributions a été de proposer pour l'animation 3D la formulation de ces intégrateurs pour une chaîne de solides articulés.
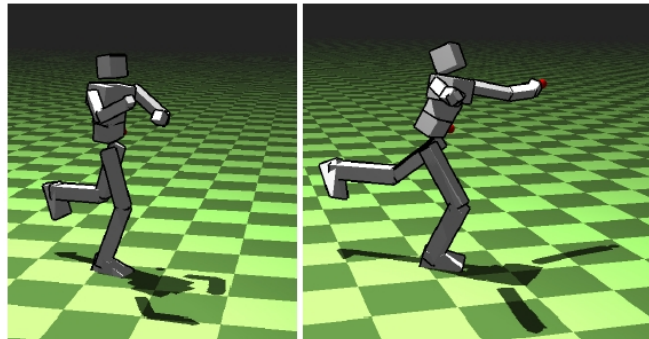
FIGURE 3.3 – *Interaction temps-réel avec un modèle physique paramétré en géodésiques principales. Dynamique, contact, équilibre et objectifs utilisateur (exprimés par la sphère rouge) sont combinés en temps-réel par optimisation quadratique.*

Cette formulation originale de la simulation a été implémentée dans un cadre d'édition de mouvement (figure 3.3). Comme pour les projets cinématiques précédents, l'interaction suit un principe de cinématique inverse. L'objectif des cibles à atteindre est formulé par un problème quadratique sur les vélocités géodésiques. En prenant l'équation de mouvement comme condition KKT d'un problème quadratique sur les mêmes

variables, on combine par pondération les deux objectifs, l'un lié aux objectifs à atteindre et l'autre lié à la dynamique. Les objectifs utilisateur peuvent être de natures cinématiques, comme un point à atteindre, ou cinétique, comme la minimisation du moment angulaire.

L'avantage de l'utilisation des géodésiques principales comme jeu réduit de variables d'état s'est retrouvé dans la possibilité d'implémenter facilement la simulation en temps réelle. De plus, la nature des composantes renforcent la robustesse dans le maintien de l'équilibre. Le système final offre à la fois une bonne interaction et un réalisme dans les mouvements secondaires. En ce sens, il tend à être adapté pour un usage en jeu vidéo. Il manque aujourd'hui à ce travail un couplage avec des contrôleurs plus avancés afin d'implémenter la locomotion.

Par rapport à un but de modélisation plus réaliste, le contrôle par objectifs engendre des forces externes, y compris sur l'articulation racine, qui ne sont pas vraisemblables d'un point de vue biomécanique. Nous avons donc commencé à étendre le système à un contexte sous actué, où seules les forces de contacts et gravitationnelles sont disponibles comme forces externes. On rentre avec cet objectif dans des problèmes liés à la gestion des contacts qui devient problématique pour le temps réel. L'usage d'un modèle physique est un formidable outil potentiel d'information a priori pour aider à la mesure de mouvement. Cependant, les choix de modélisation ici entraînaient vers une problématique de contrôle et gestion des contacts qui éloignait du thème de capture de mouvement.

3.4 MESURE DE FORCES ET DYNAMIQUE INVERSE

Sans remettre en cause la motivation à utiliser les géodésiques principales comme paramètres d'états, l'implémentation du modèle physique précédent avait conduit à une complexité trop grande pour espérer reboucler efficacement avec un objectif à plus long terme de mesure de mouvement. Le travail suivant en modélisation physique a donc été orienté sur des postulats mécaniques plus simples mais plus proche de la mesure d'information. La démarche de simplification était aussi motivée par l'étude d'un sujet particulier sur les mouvements d'escalade. En rapport avec les caractéristiques propres de ce sport, la gestion des contacts s'est réduite à des contacts bilatéraux planifiés, beaucoup plus simple à gérer que des contraintes unilatérales et une condition de complémentarité. Cette démarche a permis de mieux se focaliser sur un but d'étude de mouvement et non d'implémentation de la dynamique. L'orientation spécifique vers l'escalade découle d'une rencontre avec un candidat doctorant en informatique, intéressé par l'animation, et par ailleurs pratiquant expert de ce sport.

Une des mesures importantes en biomécanique est l'estimation des efforts musculaires aux articulations. Il était donc intéressant de l'aborder pour l'escalade. En occultant les problèmes de co-contraction, cette mesure passe par l'estimation des couples résultants aux articulations. Connaissant les grandeurs inertielles du sujet en mouvement, la mesure des forces de contacts et de leur point d'application résultant, les couples se déduisent par dynamique inverse. Pour la locomotion, cette mesure s'ob-

tient classiquement de manière déterministe avec des plateformes de force. Ces dernières sont faciles à mettre en œuvre pour la locomotion, mais sont bien sûr plus problématiques pour l'escalade. Dans un contexte informatique, nous nous sommes donc posé le défi technologique d'une estimation de la dynamique pour l'escalade, forces de contact et couples articulaires, directement à partir de données cinématiques, au mieux issues de la vidéo, au minimum issues de la capture de mouvement.

L'idée d'estimer des grandeurs physiques à partir de la vidéo avait déjà été abordée en vision par ordinateur et en animation. Dans [BSP02], sont estimés position, vélocité et orientation du champs de gravitation pour un rigide en chute libre à partir de la seule information vidéo. Plus proche de notre objectif, une méthode de prédiction de forces de contact à partir de données cinématique vidéo ou marqueurs est proposée dans [BSF09]. Cependant, ce dernier travail s'appuie sur un modèle de contact par pénalités qui s'avère certes efficace pour la détection géométrique des contacts mais n'a aucun sens physique ou métrologique.

FIGURE 3.4 – *Le mur d'escalade instrumenté, un détail d'une prise, la prédiction des forces de contacts.*

En suivant une formulation stricte de la mécanique par contraintes, et en prenant les forces de contact, de gravitation et des actionneurs aux articulations comme seules forces externes, les configurations d'un système articulé avec plus de deux points de contacts aboutissent à un problème sous-contraint. Intuitivement, étant donné un solide articulé en interaction avec l'environnement selon plusieurs points de contact, il est impossible de déterminer la répartition exacte des forces entre ces différents points de contact à partir de la seule information cinématique. La solution naturelle pour résoudre une telle indétermination est de transformer la formulation linéaire en minimisation. Typiquement, pour ce problème de mécanique sous-actuée, l'hypothèse classique est de simplement chercher à minimiser la valeur des couples aux articulations et des forces de contact. Cette idée répond à l'intuition d'une gestion optimale des efforts dans les tâches de contrôle moteur. On trouve une exploitation de cette idée pour une tâche d'animation physique à partir de poses clé en graphique [WC10] ou plus récemment en robotique pour la planification rapide de locomotion en terrain aléatoire [RBM+13] de robots bipèdes ou quadrupèdes. Nous avons donc cherché à

explorer cette hypothèse dans le cadre de l'escalade et de le comparer à une mesure validée. Nous présentons donc ici un résumé des résultats.

Une collaboration avec le Pr Kry à McGill University a permis de mettre en place l'évaluation chiffrée de l'estimation de force de contacts à partir des seules données cinématiques. Au sein de son laboratoire, le Pr Kry a construit un mur d'escalade d'environ 3m, instrumenté avec un jeu de 6 prises couplées à des capteurs de force 6D. Un système de capture de mouvement OptiTrack a permis de délivrer une information articulaire synchronisée. Ainsi, il a été possible de comparer les résultats de l'estimation des forces de contact avec la valeur réelle mesurée (figure 3.4). Des travaux similaires récents en biomécanique ont étudié une tâche de transition de posture entre la position assise et la position debout en s'aidant d'une poignée en hauteur [RCM13]. Ces travaux ont montré qu'une pondération particulière de l'optimisation quadratique entre les articulations permettait d'obtenir une meilleure prédiction des forces de contacts. Nous n'avons pas observé de phénomène similaire pour notre étude où une pondération uniforme entre tous les actuateurs donnait des résultats significativement meilleurs par rapport à la pondération suggéré par l'étude précédente.
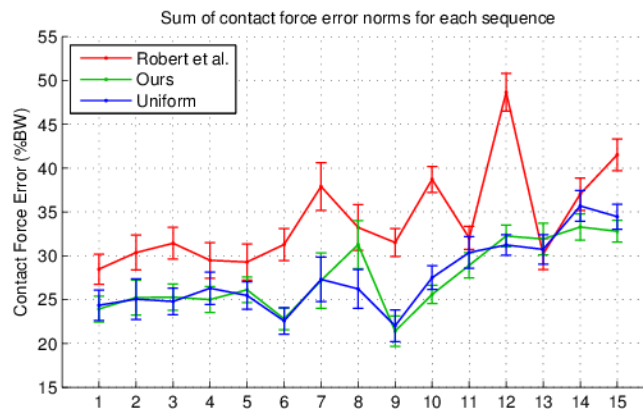
FIGURE 3.5 – *Comparaison des résultats de prédiction des forces de contacts pour 15 séquences d'escalade. En rouge, utilisation des pondérations proposées dans [RCM13], en bleu nos résultats en utilisant une pondération uniforme. En vert sont données nos résultats avec une pondération évaluée en prenant les maximums des couples observés par articulation en appliquant une première dynamique inverse utilisant les forces de contacts mesurées.*

Au final, notre prédiction des forces de contact atteint une erreur de l'ordre de 25%, ce qui correspond pour un poids de 80kg à des erreurs entre 50 et 200N (répartition de 1 à 4 points de contacts). L'amélioration est significative (p<.05) par rapport à la méthode utilisant la pondération proposée par [RCM13]. On peut y voir l'explication que la tâche d'escalade induit un recrutement plus réparti des différentes parties du corps, à l'inverse de la tâche de transition de la position assise à debout en tenant une poignée. A ce jour, plusieurs essais n'ont pas permis de faire émerger une pondération plus spécifique à l'escalade. Nous avons cependant observé que de meilleurs résultats

étaient obtenus (amélioration de l'ordre de 5 points) en limitant moins les couples aux chevilles par rapport à ce qui avait été fait dans [RCM13].

A titre d'évaluation, nous avons aussi appliqué notre méthode sur des données de marche. Nous avons pour cela utilisé les données fournies en example avec le logiciel de biomécanique OpenSim [DAA$^+$07]. Ces données consistent en une séquence classique de trajectoires articulaires et de forces mesurées par deux plateformes. Ce logiciel a permis aussi de valider nos calculs en comparant la partie finale de dynamique inverse pour le calcul de couples aux articulations. Les résultats montrent une meilleure qualité de prédiction pour les forces de contacts et les couples que pour l'escalade. Ceci était prévisible dans la mesure où le problème de contacts multiples se limite aux phases de double maintien (*double stance*) pour la locomotion. Nous avons aussi réalisé une expérience de locomotion avec une vitesse progressive. La méthode de prédiction des forces de contact est bien en mesure de capturer les changements dynamiques caractéristiques entre marche et course.
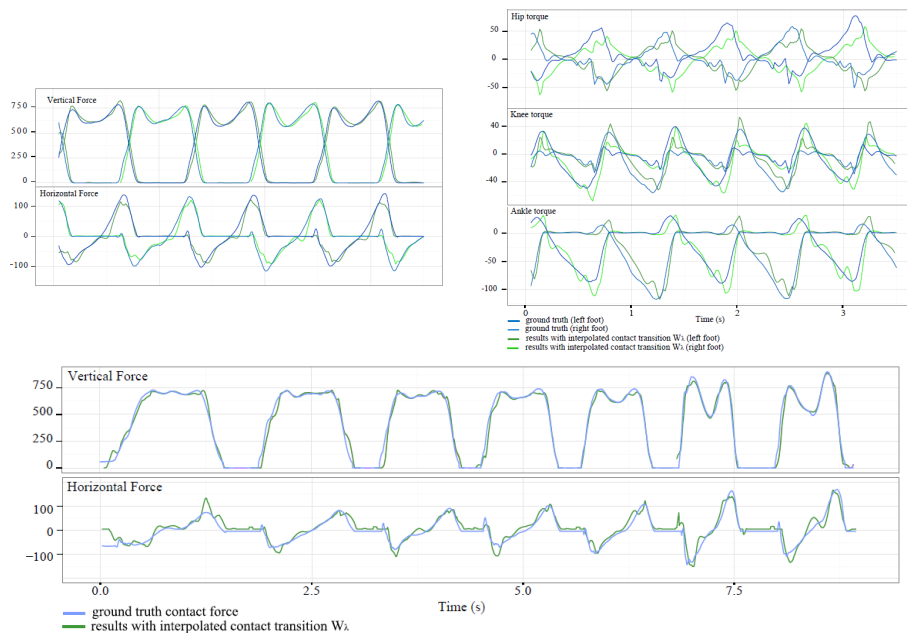
FIGURE 3.6 – *La prédiction des forces de contact (gauche) et des couples (milieu) pour une séquence de marche. Prédiction des forces de contact pour une transition de la marche vers la course (droite).*

De cet outil de mesure en dynamique inverse, nous avons déduit une application en modélisation de mouvement pour l'animation 3D. La dynamique inverse est utilisée pour optimiser le séquençage temporel (*timing*) d'une animation cinématique initiale, en calculant les vitesses d'execution de mouvement amenant à une dépense minimale d'énergie, mesurée comme la somme des couples. Dans une ascension sur un mur d'escalade, le choix des prises et leur enchaînement relève de la planification du grimpeur. Nous avons donc considéré cette information comme donnée d'entrée, déduisant

pour chaque configuration de contact une pose du corps entier par une stratégie de cinématique inverse. L'évaluation ici est d'abord visuelle. De l'interpolation à intervalles de temps constants qui sert d'initialisation à l'optimisation, le résultat présente une ascension visuellement plus convaincante. Cependant, si l'on compare ce résultat aux performances d'un grimpeur sur la même voie, on constate une différence certaine, notamment le cas réel montre une progression plus lente. Cette différence peut sans doute s'expliquer par les pauses temporelles que prend le grimpeur pour planifier cognitivement le mouvement.

CHAPITRE 4

Etude anatomique du petit animal

4.1 PRÉAMBULE

Une grande partie de mon activité de recherche a aussi porté sur le développement de systèmes expérimentaux pour le petit animal de laboratoire, souris et rat. Les tests sur souris et rats sont les premières étapes de toute mise sur le marché de médicament, de l'évaluation de la toxicité de substance chimique impliquée dans l'agro-alimentaire et de beaucoup de recherche en génétique grace au phénotypage. L'activité motrice est un indice princeps du comportement et donc sa quantification un enjeu important. Alors que le mouvement s'inscrit naturellement en trois dimensions, tous les outils actuels d'analyse automatique, dans un souci de fiabilité, ne délivrent que des informations planaires, réduisant l'animal au point 2D centroïde de sa silhouette.

La question posée était alors de savoir s'il était possible de mesurer en 3D la posture squelettique d'un rongeur à partir d'une mesure simple et reproductible. L'usage systématique de marqueurs est exclu car, outre leur difficulté ergonomique, l'hypothèse d'un couplage rigide entre le squelette et des points collés sur la peau n'est que très faiblement vérifiée. Cette question a motivé le montage du projet ANR Kameleon que j'ai coordonné. Il a rassemblé en Neurosciences l'Université Descartes-Paris5, en Anatomie le Muséum National d'Histoire Naturelle (MNHN), en Biomécanique l'Université de Bretagne et en Informatique l'INRIA. Le projet s'est notamment appuyé sur la plateforme de cinéradiographie disponible au MNHN. Le travail expérimental a donc été de la coupler avec un système vidéo multi caméra pour développer des méthodes d'analyse 3D, avec et sans marqueurs. Ce cadre expérimentale et les questions de "posturographie" du rongeur ont continuée à travers une étude du lien entre système vestibulaire et contrôle moteur, mise en œuvre en apesanteur par un projet avec le CNES. Ici encore, un équipement multicaméras spécifique a été développé pour qu'il soit compatible avec le contexte d'un vol parabolique. Enfin, un passage à l'échelle "industrielle" est en cours de développement pour l'application de la posturographie

du rongeur au problème du phénotypage rapide en collaboration avec l'Institut de Génétique et Biologie Moléculaire et Cellulaire (IGBMC), à Strasbourg.

4.2 ANATOMIE SQUELETTIQUE DU RONGEUR

L'objectif du programme de recherche sur le rongeur est d'être capable de fournir une estimation fiable de la posture squelettique 3D à partir d'une mesure sans marqueur. Cette estimation squelettique s'appuie sur la méthode d'estimation de mouvement vue au chapitre 2.2.2 qui ne fournit qu'une partie de la réponse avec le déplacement d'un maillage de référence. Pour aboutir à un encodage de la posture 3D, un squelette articulé est contrôlé via cinématique inverse par des marqueurs virtuels accrochés au maillage de référence. Ici encore, la définition de la mesure passe donc par plusieurs modèles. La section 2.2.2 présente le modèle de la forme externe contrôlée par OQR. On présente ici le squelette 3D qui fournit l'encodage finale de la mesure. L'anatomie squelettique des rongeurs est plus difficile à abstraire en une simple arborescence cinématique que pour l'humain. Nous avons donc pris le parti de commencer par un modèle anatomique détaillé pour garder la forme et l'ensemble de tous les os. C'est seulement dans un deuxième temps que le paramétrage cinématique a été simplifié comme il est fait habituellement en animation. Dans le paradigme choisi d'analyse par la synthèse basé sur l'apprentissage, la collecte des données squelettiques a alors un double but : en amont, elle va servir à fournir un jeu d'apprentissage de la forme externe (du squelette, on déduit des exemples de "cages" de contrôle paramétrées par OQR) et en aval, elle servira de données de validation auxquelles comparer l'estimation du mouvement squelettique. Ce but squelettique se décompose alors en une partie morphologique pour définir le modèle 3D et une partie mouvement.

4.2.1 Morphologie

La taille réduite des rongeurs s'adapte bien à l'analyse du corps entier par scanner. Ces scanners par tomographie délivrent un volume osseux en bloc qu'il est nécessaire de segmenter en composants mobiles pour pouvoir l'articuler. Pour l'étude de petit volume osseux, il est courant d'utiliser des scanners à tomographie X dits microCT. Ils permettent d'attendre une précision de l'ordre du micromètre sur des objets d'une dizaine de centimètre. Malgré ce niveau de résolution a priori suffisant, les premiers essais sur rat et souris ont montré cependant qu'il était difficile de distinguer clairement les frontières entre les segments osseux à articuler. Nous nous sommes donc concentrés sur des rats, animal plus gros, car nous avons pu bénéficier d'un accès à la ligne de tomographie X du synchrotron de Grenoble (European Synchrotron Radiation Facility, ESRF). Cette ligne permet d'aborder des grands volumes (le notre était de 25cmx10cmx10cm pour un rat *whistar* d'environ 200g), avec une résolution de $50 \mu m$. De plus, les caractéristiques de la source lumineuse garantissent une grande qualité d'image du fait d'une importante cohérence spatiale et fréquentiel des rayons. Les données acquises ont permis une segmentation rigoureuse des éléments osseux,

les contrastes identifiant clairement les régions intra et inter osseuses. D'une sélection des articulations principales ont ensuite été déduits les centres articulaires (figure 4.1).
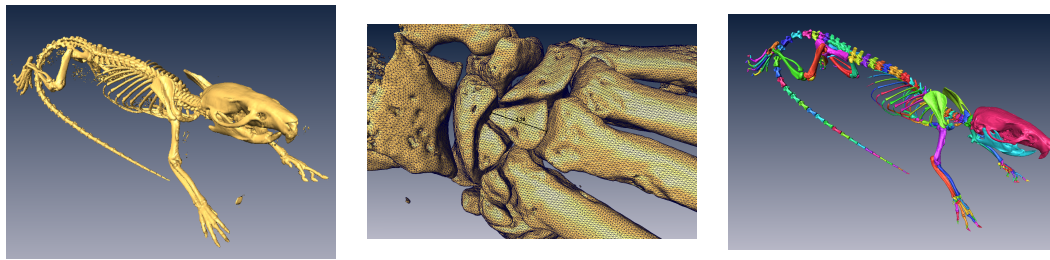
FIGURE 4.1 – *Maillage issu du volume complet acquis à l'ESRF, détail des os meta-carpiens, segmentation originale.*

Pour le rat analysé, la segmentation a été faite manuellement par des étudiants de médecine pour leur qualification en anatomie. Afin de généraliser l'approche, nous avons développé une technique de recalage automatique de tout un ensemble articulé. L'innovation a consisté à rassembler dans la même approche, non seulement une adaptation automatique de maillage articulé à la variabilité morphologique mais aussi aux changements de poses. La méthode consiste en un compromis entre déformation rigide et déformation élastique de l'ensemble des os articulés. Partant d'un modèle 3D de référence, le rat scanné à l'ESRF et segmenté par exemple, celui-ci est adapté soit à un nouveau maillage 3D non segmenté ou un volume d'images obtenu par tomographie. Si les données cibles présentent une vraisemblance suffisante avec le modèle d'origine, la convergence est assurée. Cette technique nous a permis de générer automatiquement des modèles articulés pour d'autres rongeurs, rats et souris, à partir des volumes obtenus précédemment par scanner microCT et qui n'avaient pas permis alors une segmentation efficace (figure 4.2). L'étape suivante naturelle était donc d'aborder les outils pour la capture du mouvement squelettique. 4.1).
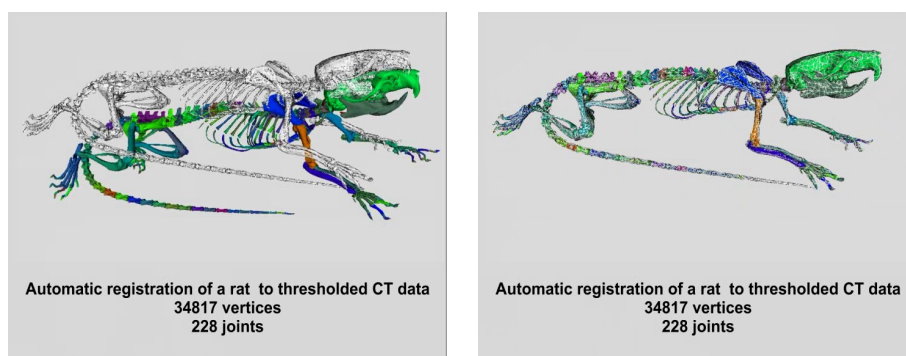
FIGURE 4.2 – *Adaptation morphologique automatique d'un modèle articulé [GRP10]*

4.2.2 Mouvement
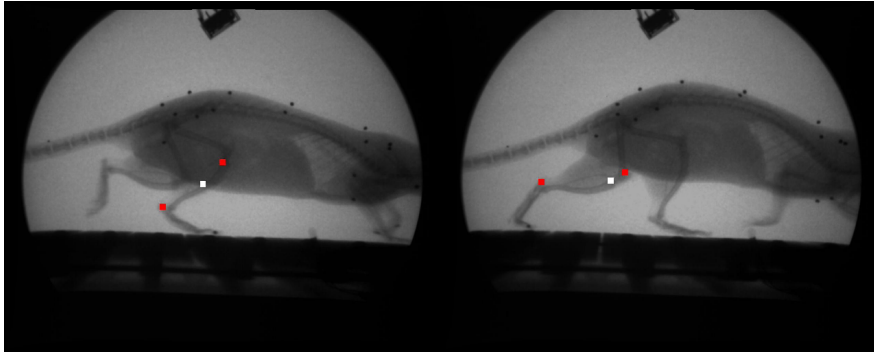
FIGURE 4.3 – *Effet de glissement de la peau : les marqueurs en rouge sont implantés sur les os par chirurgie (cheville et genou), alors que le marqueur en blanc est placé sur la peau à hauteur du tibia.*

Contrairement aux humains, les membres des rongeurs sont peu marqués. L'information de position 3D de points sur le corps se traduit plus difficilement en rotation articulaire. De plus, les expériences en cinéradiographie ont mis clairement en évidence le problème du couplage non-rigide entre le mouvement du squelette et la peau. En comparant la trajectoire de marqueurs radio-opaques insérés sur le squelette par chirurgie et la trajectoire de marqueurs collés sur la peau, il apparaît qu'à l'exception des extrémités des pattes, les articulations sous-jacentes ne sont pas facilement mesurables (4.3).
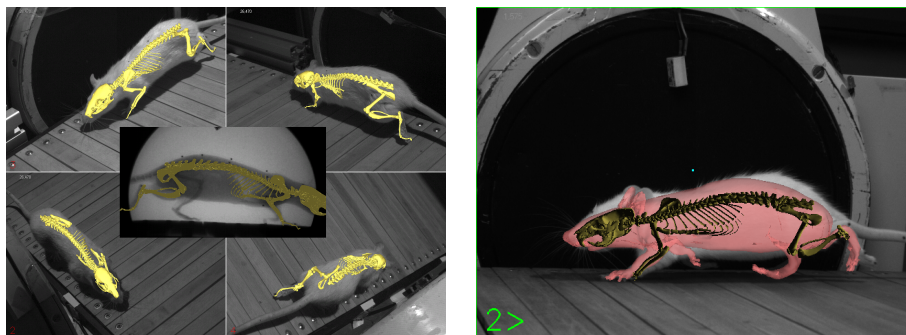
FIGURE 4.4 – *Capture de mouvement de la locomotion d'un rat, avec marqueur (gauche) et directement à partir de la vidéo (droite).*

Pour générer le mouvement squelettique à partir du peu d'information de marqueurs que nous avions, inspiré par des techniques d'animation, nous avons alors développé un système de cinématique inverse adapté à notre situation. Dans [ZVDH03], la trajectoire de marqueurs mesurés par capture de mouvement est transmise à un modèle articulé par l'intermédiaire de ressort linéaire entre les marqueurs et leur point

d'attache sur le squelette d'animation. Dans le cas du rat, en attribuant une raideur souple aux marqueurs situés sur la peau, nous avons pu compenser le faible couplage rigide entre externe et interne. Pour les marqueurs situés plus proches des structures osseuses telles que le bout des pattes, une raideur plus forte a été attribuée. La mesure 3D des marqueurs a été obtenue par calibrage et triangulation d'un système multicaméra, intégrant à la fois une vue cinéradiographique et plusieurs vues vidéos (figure 4.4).

Cette étape à base de marqueurs génère le jeu d'apprentissage pour la méthode décrite en 2.2.2. Le passage à l'analyse sans marqueurs pour le mouvement squelettique 3D consiste à placer sur le maillage de référence des marqueurs virtuels que l'on relit par le système de ressorts décrit ci-dessus. La mise en œuvre expérimentale reste lourde et peu de séquences d'apprentissage ont pu être collectées. L'apprentissage de pose n'a pu se faire que sur un ensemble très limité d'exemples et pas assez représentatif de la variabilité du mouvement. Il faudra passer à un procédé plus systématique à l'avenir. C'est une des motivations importantes pour le projet d'équipement KINOVIS décrit en 4.4, combinant à la fois vidéo multivues et cinéradiographie biplanaire.

4.3 ANALYSE POSTURALE EN 0G

Le projet précédent impliquait une équipe d'un laboratoire de Neurosciences spécialisée dans le système vestibulaire et plus largement la problématique de l'équilibre. Chez les mammifères, l'équilibre est une fonction sensori-motrice complexe intégrant action motrice, posture, et perception multimodale de l'environnement. Pour l'équilibre, cette perception implique le système vestibulaire (organes de l'oreille interne), la vision, la perception tactile des contacts avec l'environnement et la proprioception musculaire. La boucle entre action et perception est forte dans le maintien de l'équilibre puisque tout changement de posture entraîne un changement dans les signaux neurophysiologiques captés. Des mutations génétiques naturelles chez la souris peuvent conduire à l'absence totale (souche ISK) ou partielle d'organe vestibulaire (souche IED), ne se développant dans ce cas là que les canaux semi-circulaires (capteur d'accélération angulaire) et non les otolithes (capteur d'accélération linéaire). Dans les deux cas de mutation, les individus présentent des syndromes moteurs particuliers ("circling" pour les iSK, tilt de la tête pour les IED) mais les animaux adaptent globalement leur posture et locomotion pour un mode de vie normale, compensant sur les autres entrées sensorielles de l'équilibre.

En isolant certains sens, certaines variables neurophysiologiques, ces mutants apportent des renseignements intéressants sur l'organisation du système nerveux. L'idée de l'expérimentation en apesanteur vient de la volonté d'isoler encore plus de variables sensorielles pour observer, mesurer et expliquer quelle est la posture adoptée en absence d'entrées. Le recours à l'apesanteur permet de supprimer l'entrée tactile. Ne pouvant avoir recours ici encore à une quelconque technique à base de marqueur, nous nous sommes donc tournés vers la reconstruction 3D à partir d'un système multicaméra. La construction d'un tel système afin qu'il réponde aux normes de sécurité aé-
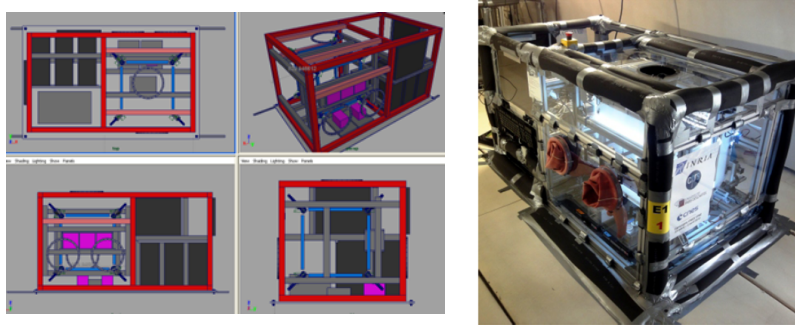
FIGURE 4.5 – *Dispositif d'acquisition multicamera 0G.*

ronautique en absence de gravité est un défi ingénieur en soi. Aux contraintes que tout le matériel d'acquisition reste d'un seul bloc en apesanteur, s'ajoute la garantie qu'aucune souris ne puisse s'échapper et vienne s'attaquer aux commandes électriques de l'avion... Le dispositif développé permet d'embarquer et tester séparément trois types génétiques de souris (sans système vestibulaire, sans otolithes et modèle contrôle). Il comprend un système d'acquisition vidéo à 8 caméras industrielles (120Hz, 640x480 pixels, monochromes), une télémétrie pour la mesure de température sur des individus n'évaluant pas vol libre et 3 PC standards pour l'enregistrement (figure 4.5).
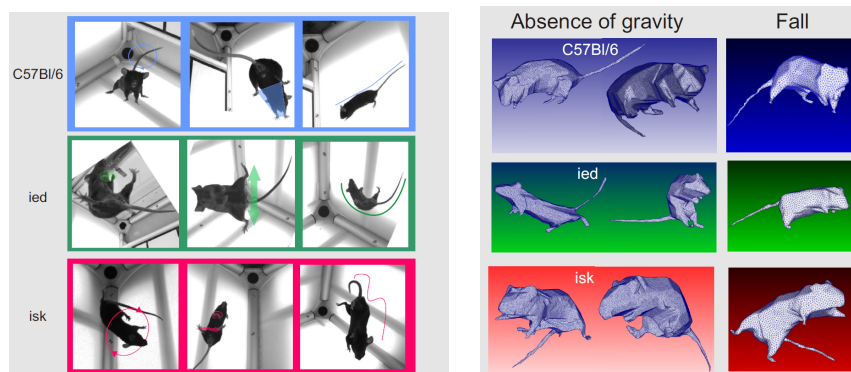
FIGURE 4.6 – *Indices posturaux qualitatifs et reconstruction 3D.*

Afin de bloquer tous les degrés de liberté des caméras, le cadrage a imposé de se limiter à un champ de vue central réduit d'environ 20cm de diamètre. Sur les 90 séances de 20 secondes en apesanteur, environ une dizaine de seconde par type de souris ont pu aboutir à une reconstruction 3D correcte par enveloppe visuelle (figure 4.6). Ces postures ont été comparées à des postures en chute libre sur les mêmes types de mutants. La dizaine de secondes par souris correspond à des cas où la souris est vue par au moins 6 caméras simultanément, permettant ainsi une reconstruction par enveloppe convexe. Les moments où elle est vue par moins de caméras pourront être exploités à l'avenir en mettant en place un système de suivi adapté. Comme suggéré lors du projet ANR

Kameleon, il faudra pour cela rassembler plus d'information a priori sur les configurations squelettiques. Ce point a conduit à un dernier investissement expérimental sur une plateforme cinéradiographique biplanaire.

4.4 LES PLATEFORMES CINÉRADIOGRAPHIQUES BIPLANAIRES

Dans la configuration du projet ANR kameleon, une seule vue cinéradiographique avait permis de trianguler en 3D la trajectoire de certains marqueurs en utilisant des caméras standards quand les marqueurs étaient visibles par celle-ci. Cette configuration ne permet pas d'accéder à la mesure 3D de marqueurs internes. Une des solutions expérimentales consiste à doubler les vues par rayons X. La cinéradiographie biplanaire permet ainsi d'estimer en 3D toute trajectoire de marqueur radio opaque implanté sur l'animal. Ce type de plateforme existe pour l'animal à l'université d'Iena (Allemagne), au Royal Veterinary College (Londres, UK) et à l'université Brown (Providence, RI) à travers le projet XROMM [BBG+10]. D'autres plateformes de cinéradiographie biplanaire sont en cours de construction, créant l'émergence d'une nouvelle communauté scientifique sur cette approche de l'étude du mouvement anatomique. Dans le cadre du projet d'équipement d'excellence KINOVIS obtenu à Grenoble, j'ai défini la spécification d'une plateforme combinant un système multicaméra à haute performance et une cinéradiographie biplanaire (figure 4.7). Cette plateforme a pour vocation non seulement d'apporter les données nécessaires pour les rongeurs, mais servira de base à une nouvelle orientation de mes activités sur l'orthopédie de la main. Rongeurs et main humaine, bien qu'apparemment très éloignés, occupe un volume anatomique comparable et présente les mêmes difficultés d'enfouissement de leur structure de contrôle. C'est en partie sur cette plateforme que s'ouvre mes nouvelles perspectives de mesure et modélisation de l'anatomie en mouvement.
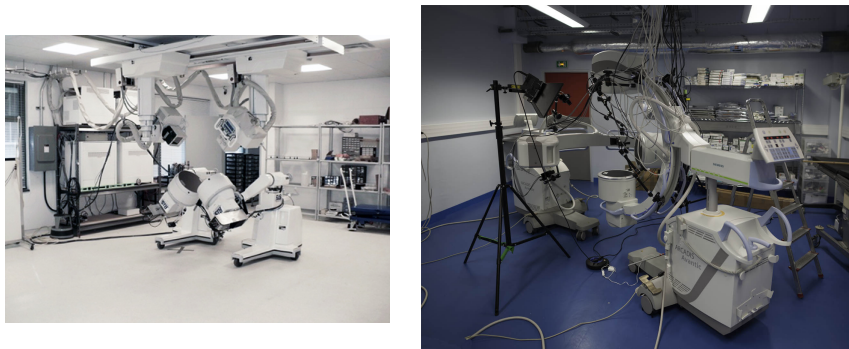
FIGURE 4.7 – *Plateformes de cinéradiographie biplanaire : à Brown University, projet XROMM (gauche) et au CHU de Grenoble, projet KINOVIS (droite).*

Articles

- Animal Gaits From Video, *L. Favreau, L. Reveret, C. Depraz, M.-P. Cani*, EG Symposium on Computer Animation, SCA'04, Grenoble, France, aug 2012.

- Motion compression using Principal Geodesics Analysis, *M. Tournier, X. Wu, N. Courty , E. Arnaud, L. Reveret*, Computer Graphics Forum, Proceedings of Eurographics'09, Münich, Germany, april 2009.

- Modal Locomotion : Animating Virtual Characters with Natural Vibrations, *P. G. Kry, L. Reveret, F. Faure, and M.-P.Cani*, Computer Graphics Forum, Proceedings of Eurographics'09, Münich, Germany, april 2009.

- Wind Projection Basis for Real-Time Animation of Tree, *J. Diener, M. Rodriguez, L. Baboud, L. Reveret*, Computer Graphics Forum, Proceedings of Eurographics'09, Münich, Germany, april 2009.

- Creating and animating subject-specific anatomical models, *B. Gilles, L. RevÃ©ret, D.K. Pai*, Computer Graphics Forum, 29(8), dec 2010.

- Natural character posing from a large database, *Xiaomao Wu, Maxime Tournier, Lionel Reveret*, IEEE Computer Graphics and Applications, 31(3), may 2011.

- Locomotion Skills for Simulated Quadrupeds, *Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, Michiel van de Panne*, ACM Transactions on Graphics, Proceedings of SIGGRAPH'11, Vancouver, Canada, aug 2011.

- Principal Geodesic Dynamics, *Maxime Tournier, Lionel Reveret*, EG/SIGGRAPH Symposium on Computer Animation, SCA'12, Lausanne, Switzerland, july 2012.

- Cage-based Motion Recovery using Manifold Learning, *Estelle Duveau, Simon Courtemanche, Lionel Reveret, Edmond Boyer*, 3DIMPVT 2012 - International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, ZÃ$\frac{1}{4}$rich, Switzerland, oct 2012.

$$\therefore$$

5.1   ANIMAL GAITS FROM VIDEO

L. Favreau, L. Reveret, C. Depraz, M.-P. Cani EG/SIGGRAPH Symposium on Computer Animation, SCA'04, Grenoble, France, aug 2012.

Animal gaits from video

Laurent Favreau, Lionel Reveret, Christine Depraz, Marie-Paule Cani

GRAVIR-INRIA

Abstract

We present a method for animating 3D models of animals from existing live video sequences such as wild life documentaries. Videos are first segmented into binary images on which Principal Component Analysis (PCA) is applied. The time-varying coordinates of the images in the PCA space are then used to generate 3D animation. This is done through interpolation with Radial Basis Functions (RBF) of 3D pose examples associated with a small set of key-images extracted from the video. In addition to this processing pipeline, our main contributions are: an automatic method for selecting the best set of key-images for which the designer will need to provide 3D pose examples. This method saves user time and effort since there is no more need for manual selection within the video and then trials and errors in the choice of key-images and 3D pose examples. As another contribution, we propose a simple algorithm based on PCA images to resolve 3D pose prediction ambiguities. These ambiguities are inherent to many animal gaits when only monocular view is available.

The method is first evaluated on sequences of synthetic images of animal gaits, for which full 3D data is available. We achieve a good quality reconstruction of the input 3D motion from a single video sequence of its 2D rendering. We then illustrate the method by reconstructing animal gaits from live video of wild life documentaries.

Key words*: Animation from Motion/Video Data, Interpolation Keyframing, Intuitive Interfaces for Animation.*

1. Introduction

Traditional motion capture methods - either optical or magnetic - require some cooperation from the subject. The subject must wear markers, move in a reduced space, and sometimes has to stay on a treadmill. The range of possible captured motions is thus very limited: capturing the high speed run of a wild animal, such as a cheetah running after his pray is totally untractable using this method. This is unfortunate since this kind of motion data would be of great interest for 3D feature films and special effects, for which fantastic animals must be animated while no source of motion is available.

The new method we propose allows the extraction of 3D cyclic motion of animals from arbitrary video sequences (we are currently using live sequences from wild life animal documentaries). State of the art techniques in computer vision for markers-less 3D motion tracking are still hard to use in an animation production framework. As an alternative, we propose to use a robust existing techniques in a novel pipeline: we combine PCA of images and animation by interpolation of examples to reliably generate 3D animation of animal gaits from video data. PCA of images is well suited for animal gaits since this motion is naturally cyclic and PCA will factorize similar patterns and isolate main variation in images. Our experiments show what constraints and additional processing can be used to help PCA to focus on coding variation due to motion only. Our goal is to isolate and characterize, using PCA images, minimal sets of cyclic motion and to subsequently generate the associated 3D animation. More complex 3D animation with non uniformly cyclic motion could later be generated using recent methods in motion synthesis. We improve existing techniques with 2 main contributions: an automatic criterion to select examples from video and an algorithm to resolve ambiguities in the prediction of 3D poses from 2D video.

The resulting method greatly saves effort for the animator. Traditionally for the animation of quadrupeds, the artist must make several trails to set the key-frames and 3D poses. Our method, based on PCA images, allows us to provide directly the visually salient key-images with which to associate a 3D pose. The interpolation methods automatically generates long sequence of 3D animation mimicking the rhythm of the original video.
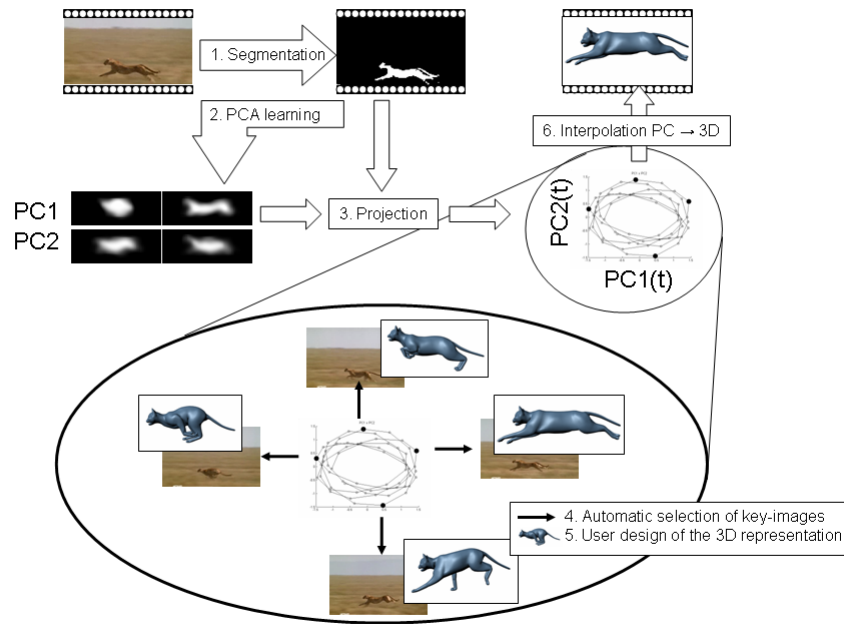
Figure 1: *Overview of the method*

1.1. Previous work

One of the first attempts to reconstruct animal motion from videos is Wilhelms's work [WG03]. Deformable contours (snakes) are used to extract the 2D motion of each limb's contour from a video sequence of a running horse. This motion is then transformed to 3D motion by matching 2D contours of limbs in the image with contours of limbs of a 3D model aligned on the image sequence. It is well known now that active contours methods are very sensitive to noise and have parameters which are difficult to tune. Wilhelms et al. [WG03] mention this problem and allow the user to reinitialize the active contours. This makes the method difficult to use in the general case, especially when limbs occlude each others. More generally, Gleicher et al. [GF02] show that current computer vision techniques for the automatic processing of videos fail to provide reliable 3D information such as stable joint angles over time. They conclude that using this kind of approach for the direct control of 3D animation at the joint level is currently not feasible.

Examples-based approaches have recently been recognized as a good alternative to traditional shape modeling and animation methods. The basic idea is to interpolate between a given set of examples, 3D pose or motion, mapped from an input parameter space to 3D data. Rose et al. [RBC98] parameterize the synthesis of new motion from motion capture data labeled with abstract parameters characterizing the style of motion. Lewis et al. [LCF00] interpolate shapes of a bending arm from joint angle values using Radial Basis Functions (RBF). They show that pose space mapping avoids well-known artifacts of traditional skinning methods. Sloan et al. [SIC01] extend this formulation by combining RBF and linear regression. All these approaches interpolate between well defined data - i.e. examples of 3D shapes or motion, labeled with user defined abstract parameters. Pyun et al. [PKC*03] show that a similar framework can be used to animate new faces from captured facial animation data. In this case, the abstract parameters are replaced by the 3D animation data that control the way the examples are interpolated over time. Visual features extracted from 2D images can also be used as input parameters to control pose space mapping. Bregler et al. [BLCD02] capture information from existing footage of 2D cartoon animation to control the blend shapes animation of 3D characters.

1.2. Overview

Our method is an example-based approach. We test video data as possible input parameters to control animation. Live video footage is challenging to process: because it lacks contrast and resolution, automatic feature extraction is not robust, and would require heavy user intervention. We rather convert the original images into normalized, binary images, on which Principal Component Analysis (PCA) is applied.

The images' coordinates in the Principal Component space provides an adequate set of parameters to control the 3D motion.

When input parameters are derived from a large set of data, all examples-based methods require that the user explicitly designate the examples. We propose a new and automatic criterion for selecting these examples. Radial Basis Functions (RBF) are used to interpolate between these pose examples over time, from the sequence of parameter values extracted from the video.

Section 2 presents our general pipeline for generating 3D animation from video: it details the chain of operations that we apply to the video sequences in order to extract adequate control parameters, and the way we interpolate given 3D pose examples to generate the animation. In particular, the conversion to binary images can either be fully automatic or use simple user input such as rough strokes sketched over the images: We show that both methods provide similarly good data for applying PCA.

Section 3 presents two extra contributions. First, we present a criterion for automatically selecting the best, minimal set of key-images from the video-data. Providing such a criterion prevents the user from spending hours carefully analyzing the input motion in order to find out which images he should associate with 3D pose examples. Second, we propose a simple algorithm to resolve ambiguities in the prediction of 3D pose from 2D silhouettes.

We validate our method in Section 4, by testing our approach on synthetic data: as our results show, we achieve a precise reconstruction of existing 3D animations of animal motion from video sequences of their 2D rendering, given that the right 3D shapes were associated with the automatically selected poses.

Section 5 presents our final results: wild animal motion is extracted from real life documentaries. Several features of our method, such as the option of filtering the coordinates in Principal Component space before applying the interpolation are discussed. We conclude and give directions for future work in Section 6.

## 2. Predicting 3D animation using PCA on images

2.1. Overview of the method

Our approach combines statistical analysis of binary images by PCA and animation by pose space mapping. The binary images can be generated by automatic segmentation. When automatic segmentation fails, we propose a sketching tool to label the video. In this case, white strokes on a black background create the binary image. PCA is then applied on the binary images, taking each image as a single observation vector. The projection coefficients of input images onto the Principal Components are analyzed to extract optimal

examples of 3D poses to interpolate. These projection coefficients serve as input parameters to the pose space mapping interpolation and control the temporal evolution of the animation (Figure 1).

2.2. Reducing variability into binary images

Using PCA directly on images would encode any variation in appearance. In addition to variation due to motion, changes in illumination, camera motion and occlusion would be coded as well by PCA. Thus, before applying PCA, video images are segmented into binary images in order to filter such variation and isolate the foreground subject from the background. Assuming the user can provide some initial guess on the subject and background location on the first image by selecting two rectangular areas for each one on the first image of the sequence, a simple segmentation based on mixture of Gaussians can still provide accurate results (Figure 1 and top of Figure 2). This method is easy to implement and was sufficient for the purpose of our work on gaits generation. More elaborated techniques could be used and provide even more accurate input data to our approach [SM00].
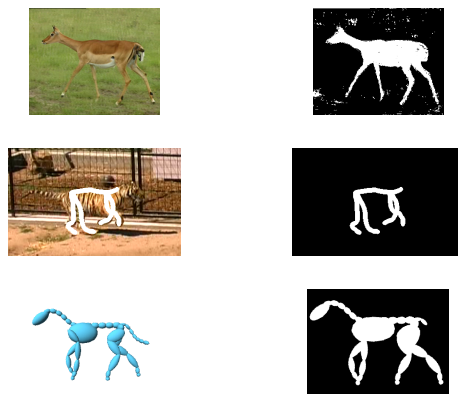
Figure 2: *Results of segmentation for our three sources of data: live video, sketching and synthetic*

When automatic segmentation fails, we propose to the user a sketching interface to label the video footage. The sketches does not need to be accurate as in Davis et al.[DAC*03], where the drawing needs to be precise enough so that the joints can be automatically recognized. In our case, the huge change in illumination and high occurrence of occlusions make impossible to claim for a careful joint to joint labeling. Instead, we rely on a raw labeling with strokes of the main features such as the spine, legs and head. It is not required to label every joint individually if they don't appear in the image. The idea is to similarly apply a PCA on images, either generated from segmentation or resulting from sketching.

Once the subject is isolated from the background, a region

of interest is automatically extracted around the silhouette by standard morphological analysis and detection of connected components. This process is applied to all the images in the video sequence. We keep track of the center of mass of the binary silhouette evaluated at the previous image so that the region of interest is still focussed on the correct connected component.

This step allows us to get rid of variance due to camera motion which is not relevant to the true motion of the tracked subject. Unfortunately, it also filters out the vertical translation of the animal, which is relevant to motion. Nevertheless, we are not trying to extract the translation as an independent parameter. Instead, our aim is to capture the overall timing and predict animation by interpolation of 3D pose examples. Consequently, if a vertical translation is set in the pose examples, assuming such a motion is correlated with the rest of the visible motion in the images sequence, it will appear in the final animation. Typically, a full body extension is correlated with a flight part in the gait scenario.

From this pre-process, we end up with a sequence of binary images. We give here the data specifications for our 7 test sequences in terms of number of frames, size in pixels of the original image, and size in pixel of the tracked window of the silhouette.

Sequence	Number of frames	Original image	Binary image
Horse walk	100	320x240	320x240
Horse canter	100	320x240	320x240
Horse gallop	100	320x240	320x240
Cheetah run	137	192x144	90x34
Tiger run	60	720x480	448x216
Antelope walk	122	352x288	232x173
Giraffe walk	73	352x288	195x253

2.3. Principal Components as input visual features

Principal Components Analysis (PCA) is a standard method for data reduction. It consists in finding a set of principal axes, linearly spanning the variance of multidimensional data. Turk and Pentland introduced one of the first implementations of PCA on images to perform face recognition (eigen-faces) [TP91]. In this case, each image is considered as an independent observation where all the pixels values are stacked in a single vector. Eigen-images have been widely used to reduce, classify and recognize regular patterns from images. As a new contribution we show that PCA on images can encode variation due to motion only and can be used not only to classify shapes but also to continuously predict change in motion. We will take benefit of this property in the interpolation scheme.

PCA consists in calculating the eigenvectors and eigenvalues of the covariance matrix of the collected data. In our case, each rectangular image of the sequence is viewed as a

row vector $\mathbf{i}(t)$ of all the pixels values stacked together. We gather all the n images over a sequence in a matrix $\mathbf{I}$, after having subtracted the mean image $\bar{\mathbf{i}}$:

$$\bar{\mathbf{i}} = \frac{1}{n} \sum_{t=1}^{n} \mathbf{i}(t) \qquad (1)$$

$$\mathbf{I} = \left[\left(\mathbf{i}(t_1) - \bar{\mathbf{i}} \right)^t, \ldots, \left(\mathbf{i}(t_n) - \bar{\mathbf{i}} \right)^t \right]^t \qquad (2)$$

The PCA is then formulated as:

$$\frac{1}{n} \mathbf{I}^t \mathbf{I} \mathbf{E} = \mathbf{E} \mathbf{D} \qquad (3)$$

$$\mathbf{E}^t \mathbf{E} = \mathbf{1} \qquad (4)$$

Finally, we take as input vector of the animation the projection coefficients onto the Principal Components stacked as column vectors in matrix $\mathbf{E}$ and normalized by the square roots of the eigenvalues stacked in the diagonal matrix $\mathbf{D}$:

$$\mathbf{p}(t) = \left(\mathbf{i}(t) - \bar{\mathbf{i}} \right) \mathbf{E} \sqrt{\mathbf{D}^{-1}} \qquad (5)$$

We recapitulate below the results of PCA in terms of part of the variance covered by each Principal Component with respect to the total variance of the data for our 7 test sequences.

Sequence	PC1	PC2	PC3	PC4
Horse walk	33.7	23.7	11.4	8.56
Horse canter	32.5	14.5	9.17	8.78
Horse gallop	31.1	19.9	11.0	8.33
Cheetah run	44.7	11.6	9.93	7.79
Tiger run	15.2	10.5	6.14	4.69
Antelope walk	21.5	12.2	8.40	6.91
Giraffe walk	42.8	15.8	11.1	5.63

2.4. Interpolation

Our goal is to generate animation parameters (position and joint angles) $\mathbf{x}(t)$ from the values of projection coefficients $\mathbf{p}(t)$ computed from PCA. We use interpolation of m 3D pose examples $[\mathbf{x}(t_i)]_{i=1\ldots m}$, corresponding to m images in the video sequence for which we know the projection coefficients $[\mathbf{p}(t_i)]_{i=1\ldots m}$ at time t_i in the video sequence. For clarity, we note $\mathbf{x_i}$ and $\mathbf{p_i}$ for respectively $\mathbf{x}(t_i)$ and $\mathbf{p}(t_i)$.

Three main methods for scattered data interpolation are used in example-based method approaches: linear interpolation[BLCD02], Radial Basis Function [LCF00] or a combination of both[SIC01]. In the latter case, linear interpolation allows us to cope with cases where input data could be sparse and require a stable behavior for extrapolation. In our case, input data is the results of PCA and as such is already linearly compact. For this reason, Radial Basis Function (RBF) were enough to deal with our case. This

general interpolation scheme is formulated as linear combination of distance functions $h(r)$ (the RBF) from m interpolation points in the input space:

$$\mathbf{x}(\mathbf{p}) = \sum_{k=1}^{m} h(\|\mathbf{p} - \mathbf{p_k}\|)\mathbf{a_k} \tag{6}$$

where $\mathbf{p}$ is the input vector and $\mathbf{x}$ the predicted vector. $h(r)$ are the RBF. $\mathbf{a_k}$ are unknown vectors to be determined. If the RBF are stacked into a single vector $\mathbf{h}(\mathbf{p})$ and the unknown coefficients $\mathbf{a_k}$ as row vectors into a matrix $\mathbf{A}$, we have the formulation :

$$\mathbf{x}(\mathbf{p}) = \mathbf{h}(\mathbf{p})\mathbf{A} \tag{7}$$

$$\mathbf{h}(\mathbf{p}) = [h(\|\mathbf{p} - \mathbf{p_1}\|)\dots h(\|\mathbf{p} - \mathbf{p_m}\|)] \tag{8}$$

$$\mathbf{A} = \left[\mathbf{a_1^t}, \dots, \mathbf{a_m^t}\right]^t \tag{9}$$

As interpolation points, we use m 3D pose examples $\mathbf{x_i}$ and the the values of the m associated input parameters $\mathbf{p_i}$ of the corresponding key-image. $\mathbf{A}$ has to be solved so that $\|\mathbf{x}(\mathbf{p_i}) - \mathbf{x_i}\|$ is minimal. This minimization in a least square sense leads to the standard pseudo-inverse solution :

$$\mathbf{A} = \left(\mathbf{H^t H}\right)^{-1} \mathbf{H^t X} \tag{10}$$

where,

$$\mathbf{X} = \left[\mathbf{x_1^t}, \dots, \mathbf{x_n^t}\right]^t \tag{11}$$

$$\mathbf{H} = \left[\mathbf{h}(\mathbf{p_1})^t, \dots, \mathbf{h}(\mathbf{p_m})^t\right]^t \tag{12}$$

The final formulation is then :

$$\mathbf{x}(\mathbf{p}) = \mathbf{h}(\mathbf{p}) \left(\mathbf{H^t H}\right)^{-1} \mathbf{H^t X} \tag{13}$$

Note that this can be re-formulated exactly as an interpolation of the $\mathbf{x_i}$:

$$\mathbf{x}(\mathbf{p}) = \sum_{i=1}^{m} w_i(\mathbf{p})\mathbf{x_i} \tag{14}$$

by extracting the matrix $\mathbf{h}(\mathbf{p}) \left(\mathbf{H^t H}\right)^{-1} \mathbf{H^t}$. In [AM00], Alexa et al. compress and animate 3D sequences from principal components (PC) learnt on a fully available sequence of 3D data. In our case, PC are learnt from image space and animation of 3D data is controlled by interpolation.

The value of $\sum_{i=1}^{m} w_i(\mathbf{p})$ should stay close to 1 to guarantee that a point $\mathbf{p}$ in the input space is close enough to interpolation points and any $w_i(\mathbf{p})$ should be close to $[0, 1]$ so that $\mathbf{x}(\mathbf{p})$ stays close to the convex hull of the 3D pose examples.

For the choice of $h(r)$, a common practice is to use a gaussian function for its $\mathcal{C}^\infty$ continuity properties:

$$h(r) = e^{-\alpha r^2} \tag{15}$$

The parameter α in equation 15 needs to be determined. Statistically, projections on PC are homogenous with standard deviation. This means data will be spread approximatively in every projected direction over the same interval $[-1; +1]$ - varying according to the nature of distribution. Assuming interpolation points are well spread, we take a value of 2 as a raw estimate of the distance between interpolation points. At midpoint between two interpolation points, we expect an equal influence. This can be translated into the fact that we want $h(r)$ to be equal to 0.5 when $r = 1$. This leads to an estimate of $\alpha = \ln 2$.

All previous works on example-based animation rely on the user to decide where 3D pose examples need to be provided [LCF00, SIC01, PKC*03]. In our case, this would mean selecting key-images among thousands of a video sequence. Given the number of key-images to provide, we present an automatic criterion to select these ones within the video sequence.

3. Key-images selection

3.1. Criterion for automatic selection

We want smooth mapping between the image space and the animation space as we based all our timing control on images. A small change in the image space must produce a small change in the animation space. We notice that the interpolation scheme on RBF involves the inversion of a matrix $\mathbf{H^t H}$, build from the interpolation points, as it has been shown in the previous section. Consequently, to ensure a stable interpolation, and thus a smooth animation, *we select key-images over the sequence which minimize the condition number of the matrix $\mathbf{H^t H}$ to invert*. The condition number is evaluated as the ratio of the largest singular value of the matrix to the smallest. Large condition numbers indicate a nearly singular matrix.

This criterion is generally applicable to any example-based method. It can be used to select any number of input examples, key-images in our case, when they have to be chosen within a large set of data. The singular values of $\mathbf{H^t H}$ are the squared singular values of $\mathbf{H}$. This matrix measures the respective distances between the interpolation points. Intuitively, the criterion on condition number thus selects input examples which are equally spread within the data set. Having all the singular values closed to each other means they equally sample every direction of the input space.

In practice, as will be shown in section 4 and 5, only few principal components and few 3D pose examples are needed. This allowes us to implement a simple combinatory approach for the condition number criterion: for each sequence of n frames, given a number of c principal components to consider and a number m of key-images to select, we evaluate the condition number of all the $\binom{n}{m}$ matrices $\mathbf{H^t H}$. The $\mathbf{H^t H}$ matrix is square and its dimension is m. We keep

the set of *m* key-images within the whole sequence providing the $\mathbf{H^tH}$ matrix having the smallest condition number. Keeping only a few Principal Components makes the computation fast. We tested with up to 5 Principal Components, but experiments showed that 2 were enough as will be detailed in following sections.

As an example, for the prediction of 3 sequences of animation from synthetic images, we plot the projections on the two first components as a 2D graph and search for the best 4 examples based on the condition number criterion (next section will show that 2 PC and 4 keys is the best configuration for the prediction of this specific gait). In this case the condition number criterion has the particularity to select examples at approximatively the extreme variation of the two first PCA projections (Figure 3).
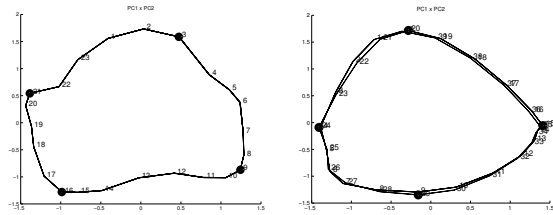
**Figure 3:** *PC1(t)xPC2(t) : PC projections across time for two synthetic sequences of horse: canter and walk. Frames are numbered for one cycle. Circles are selected examples by the condition number criterion.*

Intuitively, the more key-images are given, the better the interpolation will be. As any key-image will require the user to provide a 3D pose example, a compromise must be found. The question of the number of key-images needs to be examined on a case-to-case basis. From our experiments on animal gaits, we observed good results with 4 pose examples for the running cases and 8 pose examples for the walking cases.

3.2. Resolving 2D ambiguities with switching models

At this point, our method predicts 3D motion from silhouette images. It results in a unique 3D pose for each distinct input image. In some cases however, two different 3D poses can lead to very similar silhouettes when viewed from the side (Figure 4). This is very common in motions that consist in a succession of two symmetric phases, such as quadrupeds walking. The motion predicted by RBF still provides good results but only on one half of the period of the original gait.

To avoid this problem, it is first necessary to provide two different 3D pose examples for each of the ambiguous silhouette of the key-image and secondly to build a method to correctly choose between these two poses during the generation of the 3D animation.
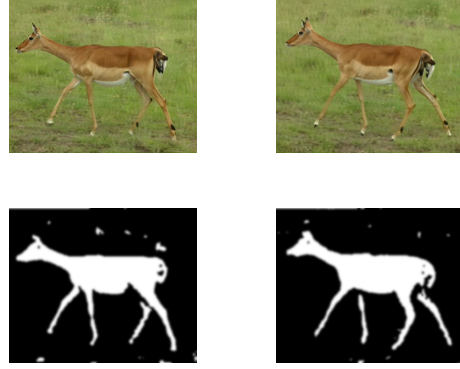
Figure 4: *Two different poses can produce similar silhouettes.*

We solve for the first problem with a simple algorithm:

1. We select *m* initial key-images with the standard method and build the animation by associating 3D poses to key-image and using RBF prediction. If the user acknowledges issues about pose ambiguities, we go to step 2.
2. For each key-image, we automatically search for its closest image in the PCA space and propose it to the user as the alternative pose for this silhouette. We constrain this image to be at least 3 frames further than the initial key-image to guarantee that we are in another half-cycle.
3. When the user validates the proposed image as the key-image corresponding to the same silhouette but at a different pose, we ask the designer to provide the appropriate 3D pose example.
4. We iterate until each of the *m* initial key-images of step 1 has its associated key-image corresponding to the opposite pose.

At the end of this process, we have doubled the number of *m* initial key-images and corresponding 3D pose examples. Figure 5 provides an example for this algorithm with $m = 4$ initial key-images. We are able now to generate a full cycle of motion. To generate animation, the same method of prediction from images is kept, but instead of keeping the same *m* 3D pose examples, we switch between *q* sets of *m* 3D pose examples as time evolves, taken from the 2*m* 3D pose examples selected by the previous algorithm. We call these *q* sets the switching models. The prediction of animation parameters is extended as follows :

$$\mathbf{x}(\mathbf{p}_t) = \sum_{k=1}^{q} w_{\sigma_k(s_t)}(\mathbf{p}_t)\,\mathbf{x}_{\sigma_k(s_t)} \qquad (16)$$

$$s_t = switch(\mathbf{p}_t, s_{t-1}) \qquad (17)$$

where s_t represents a phase state index in term of switching model, $\mathbf{x_i}$ the 2*m* pose examples, and w_i the model weight given the input image and the current phase state. The function $switch(\mathbf{p}, s)$ indicates which set of *m* pose examples

needs to be used in the prediction algorithm. It is a discrete state variable, incremented each time we detect that we have reached the last 2D silhouette within a set of m key-images. The change of silhouette in key-images is easily detected by a distance function in the PCA space.
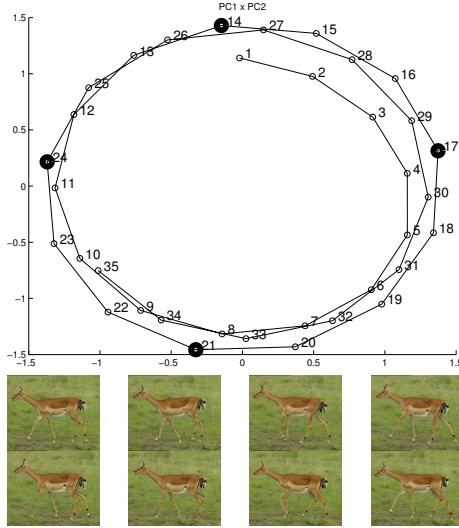
Figure 5: *Selecting more key-images to resolve ambiguities in the PCA space. Images 14, 17, 21 and 24 are selected as initial key-images. In a second step, 27, 4, 8 and 11 are selected as candidates for ambiguous pose, based on their coordinates in the PCA space. First row : frames 14, 17, 21 24; Second row : frames 27, 4, 8, 11.*

Switching between $q = 2$ models of m pose examples would allow to explore the whole animation space, as we have just doubled the number of initial m pose examples. However, the transition between two models turned out to be unstable. We solved this problem by introducing overlaps between intermediate models. The use of $q = 4$ switching models allows smooth transitions. In practice we use $m = 4$ pose examples to describe half of a cycle motion. The function $\sigma_k(s)$ gives the 4 indices of pose examples at 4 state positions with 2 overlapping pose examples between two consecutive steps:

$$\sigma_i(j) = \sigma_{ij} \tag{18}$$

$$(\sigma_{ij}) = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 4 & 5 & 6 \\ 5 & 6 & 7 & 8 \\ 7 & 8 & 1 & 2 \end{pmatrix} \tag{19}$$

4. Validation on synthetic images

We have validated our method by taking as input images the rendering of a skeleton horse 3D model. Joints are represented as ellipsoids (Figure 2). The choice of such a model

was made to get rid of any bias that a skinning algorithm would introduce. We report results for three sequences: gallop, canter and walk. By using synthetic images, we can still test the full pipeline as described in section 2. In addition, we can compare with the original animation parameters. This evaluation gave use hints on the number of principal components and examples that should be used.

We have exhaustively evaluated the results using an increasing number of key-images (starting at two) and an increasing number of PC (starting at one). Given the number of key-images to select from the video, the condition number criterion tells what key-images to select. The corresponding 3D pose examples are provided by the original animation sequence. We evaluate the results by computing the mean (and standard deviation) of the absolute difference over all the joint angles for the main rotation axes (perpendicular to the image plane, 36 angles in the case of our model) between original and predicted values.
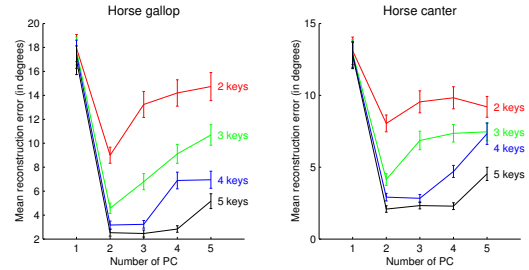
Figure 6: *Evaluation for the gallop and canter sequences. Each curve corresponds to the mean error with a fixed number of examples (2 to 5), with respect to the number of components used as input parameters.*

From Figure 6, we immediately observe that adding the third and following components introduce noise. This suggests they are coding information not relevant to the gait motion. As for the number of keys, as expected, the more examples are provided, the smaller the error. A good compromise arises on 4. Adding a fifth keys decreases the mean error less than a degree. The results are confirmed on the video provided with this paper. Two or three pose examples, although optimally selected by the condition number criterion, are not enough. With four 3D pose examples and two Principal Components, we obtain a very good match between the original animation and the predicted animation from images.

5. Processing live video sequence

We discuss now how to apply our approach on live video images, sometimes emphasized by a rough sketch as mentioned in section 2. As detailed below, strictly focussing on the first two PC and applying a band-pass filter to the PC trajectories along time enables us to achieve as good visual results as with the synthetic data.

5.1. Restricting to the two first PC

In the case of the synthetic examples, the first two components exhibit consistently interpretable behavior. For example, for the gallop of the horse, The first component (PC1) encodes a variation between a flight phase, when none of the feet touch the ground, and a grouped phase. The second component (PC2) corresponds to an opposition between a rising phase, when the horse jumps off the ground, and a descending phase, when the horse front feet hit the ground (Figure 7).
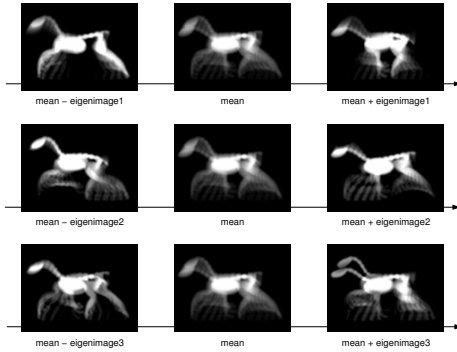
Figure 7: *Variation encoded by the first three eigenvectors for the horse gallop sequence. Middle column is the mean shape, each row corresponds to the variation along an eigenvector.*

Numerical evaluation on synthetic images have suggested that the two first PC are optimal to achieve good prediction. Image segmentation and sketching by hand will naturally introduce more noise in PC curves, making PC unstable and poorly reliable to predict relevant motion. We decide from these observations that the only 2 first PC should be kept for live video and sketched images.

We confirm this hypothesis on the cheetah sequence where similar interpretation as that for the horse gallop can be made on the two first principal components (Figure 8).
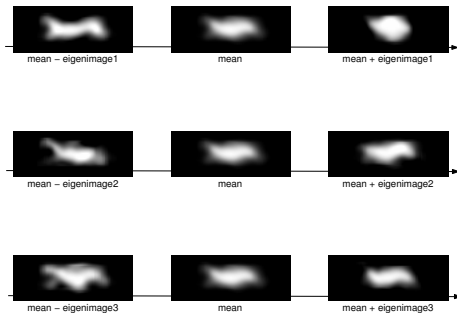
Figure 8: *Variation encoded by the first three eigenvectors for the cheetah sequence.*

5.2. Spectrum regularization

On the synthetic sequence, the time variation of the projections on the first two components shows a shift in phase of one fourth of the cycle period, corresponding to an alternation of jump, flight, landing and grouping legs. This produces the circular pattern shown on Figure 3. This configuration has been reproduced on every examples of synthetic images. Consequently, we adopt the configuration of projections on PC1 and PC2 in a circular pattern as a characterization of a video sequence to be usable with our method. In the Fourier domain, this configuration corresponds to peaks at the same location for projections on PC1 and PC2, and a phase difference of approximatively $\frac{\pi}{2}$.

Live video can thus be diagnosed as not usable by our method if it does not have projections on PC1 and PC2 staying within a certain bandwith that we automatically estimate. The first component encodes most of the variance and is considered to be representative of the fundamental cyclic variation. Its spectrum will thus be centered around a frequency corresponding to the period of the cycle. All our experiments confirm this hypothesis. From a Fourier Transform we get the frequency of maximum amplitude. We fit a peak function centered on this frequency of the form:

$$peak(f) = \frac{1}{1 + (\frac{f - f_0}{f_b - f_0})^2} \qquad (20)$$

f_0 is set at the frequency of maximum amplitude and f_b is set so that it corresponds to the closest frequency to f_0 having an amplitude of half of the maximum. We deduct a bandwidth of $[-3(f_b - f_0); +3(f_b - f_0)]$, corresponding to end points at 10% of the maximum amplitude (Figure 9).
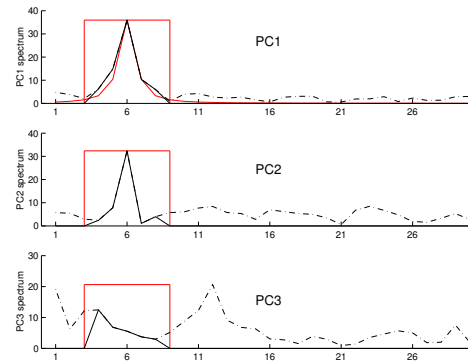
Figure 9: *Spectrum of the 3 first PC of the cheetah spectrum. The peak function is fitted to PC1, and a rectangular window is deduced.*

The second component is filtered by the same band-pass filter. What is expected is that the second component shows a similar peak, creating the circular pattern. We can evaluate how this hypothesis is respected by comparing how

much the reconstructed signal after filtering, matches the original signal. For this, we compute the correlation coefficient between the original principal component signal and the filtered principal component signal. We have observed that for our test sequences, the two first components shared the same peak ($r \geq .6$), while the following components do not ($r \leq .3$), see table below. This provide a numerical criterion to evaluate if our method can be successfully applied to a video sequence as we have presented it.

| PC:	PC1	PC2	PC3	PC4	PC5
horse gallop	.90	.92	.01	<.01	<.01
horse canter	.94	.89	.08	<.01	.01
horse walk	.99	.94	<.01	.05	<.01
cheetah run	.91	.61	.14	.21	.25
antilope walk	.78	.91	.14	.10	.06
tiger run	.87	.72	.26	.18	.23
giraffe walk	.92	.82	.24	.19	.28

Figure 9 shows the results in the Fourier domain for the cheetah sequence which conforms to the criterion. When the live video sequence fails to conform to this criterion, we suggest using the sketch approach. If the sketch approach still fails to meet the criterion, we diagnose that our method cannot work on the analyzed video.

5.3. Results

We show in the video provided with the paper results for a cheetah run, an antilope walk (both automatically segmented) and a tiger run and a giraffe walk where automatic segmentation fails but sketch images succeed.

Figure 10 shows the evolution of the weights of the four 3D pose examples for the cheetah sequence. We have an exact interpolation at these pose examples. For the rest of the sequence, we observe a correct generalization, the influence of each pose example appears at a right pace in a coherent order. Note that the sum of weights stays close to one, guaranteeing that the input parameters are always close to an interpolation point. The weights are sometimes outside of the range of $[0, 1]$ as they are not constrained in the RBF formulation. This lets the resulting pose leave the convex hull of the pose examples. This flexibility allows some extrapolation in 3D space introduced by image variations along the sequence. A control could be easily added to maintain these weights within a safe range in order to avoid the generation of strange pose, too far outside of the convex hull of the pose examples.

Finally, Figure 11 gathers the final results about key-images selection. It shows the automatically selected key-images and the associated 3D pose provided by the artist. The full video of all the tested sequences are given in the demo movie file.
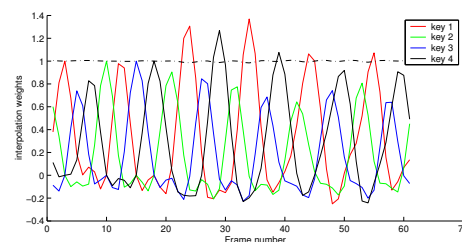
Figure 10: *Evolution of the weights of the 4 examples for the cheetah sequence. The dashed line is the sum of weights.*

6. Conclusion

When traditional motion capture of non-cooperative subjects such as wild animals is not feasible, live video footage can still provide significant information on motion. This paper has proposed novel and robust tools to analyze video data and make it applicable to the generation of 3D motion. We rely on Principal Component Analysis to extract parameters from binary version of the input images. As our result show, a small number of parameters is sufficient for cyclic animal gaits: using the two principal components already gives good results. We provide a criterion for selecting the best set of key-images from the video. In our application, the selected poses can easily be interpreted in terms of extremal images in the 2D Principal Component space.

Our work shows that Principal Component Analysis (PCA) can be applied onto a sequence of 2D images to control 3D motion. PCA on images helps to give a quantification of the significant changes in the appearance of the video. The RBF interpolation of pose examples aims at transposing the pace of video changes into the animation domain. The automatic selection of examples helps to focus the effort of the designer on the most important key-frames.

As a future work, we are planing to explore non uniformly cyclic motion such as transition between gaits and the addition of physically-based constraints to animate non-cyclic part of the motion. We are also studying how to re-use existing PCA basis and its 3D associated poses to automatically analyze a new video sequence thanks to morphological adaptation in the image space.

References

[AM00] ALEXA M., MÜLLER W.: Representing animation by principal components. In *Proc. EUROGRAPHICS'00* (2000).

[BLCD02] BREGLER C., LOEB L., CHUANG E., DESHPANDE H.: Turning to the masters: motion capturing cartoons. In *Proc. SIGGRAPH'02* (2002).

[DAC*03] DAVIS J., AGRAWALA M., CHUANG E.,
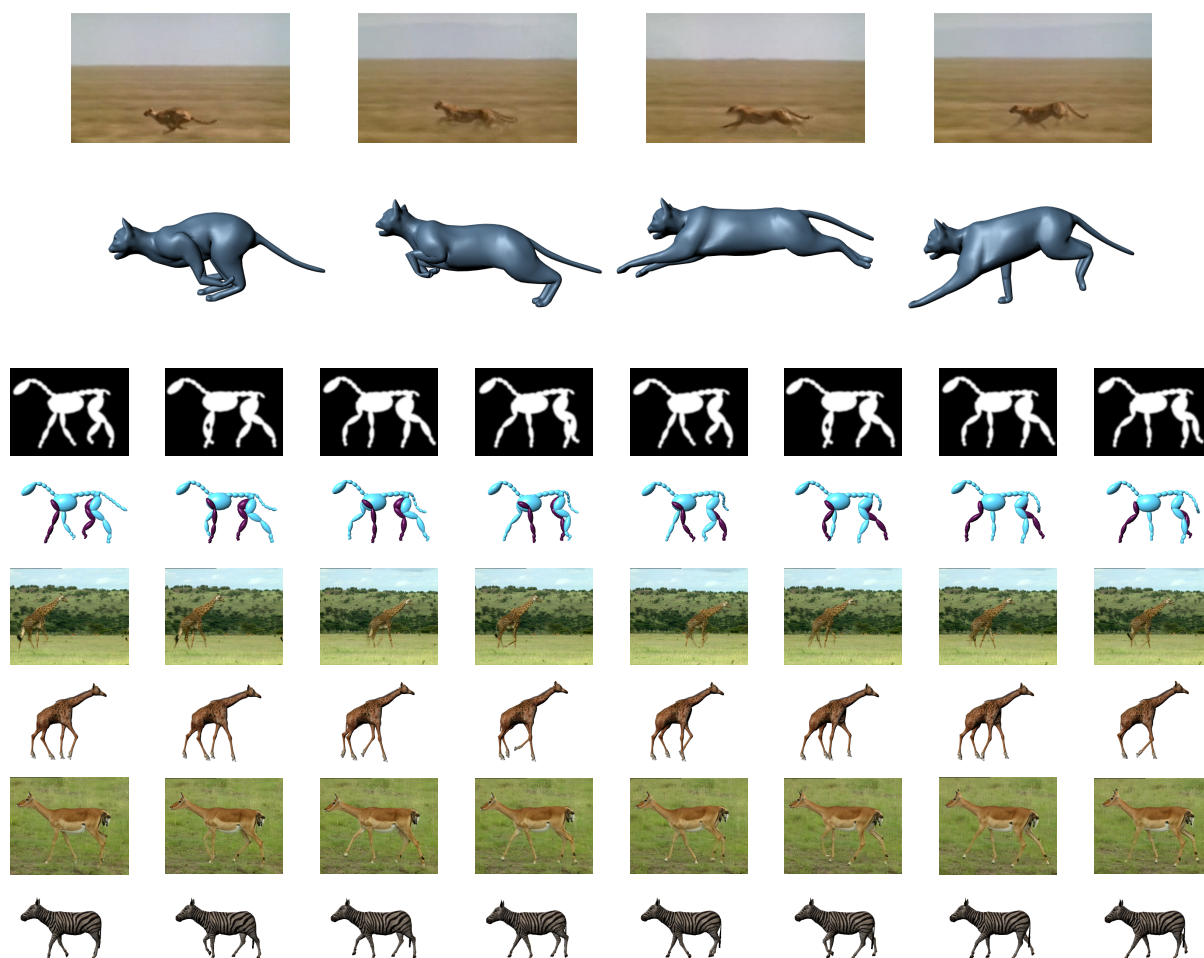
Figure 11: *Selection of key images from video sequences.*

POPOVIC Z., SALESIN D.: A sketching interface for articulated figure animation. In *Proc. EG/SIGGRAPH Symposium on Computer Animation, SCA'03* (2003).

[GF02] GLEICHER M., FERRIER N.: Evaluating video-based motion capture. In *Proc. of Computer Animation, CA'02* (June 2002).

[LCF00] LEWIS J., CORDNER M., FONG N.: Pose space deformation: A unified approach to shape interpolation and skeleton-driven defomation. In *Proc. SIGGRAPH'00* (2000), pp. 165–172.

[PKC*03] PYUN H., KIM Y., CHAE W., KANG H. W., SHIN S. Y.: An example-based approach for facial expression cloning. In *Proc. EG/SIGGRAPH Symposium on Computer Animation, SCA'03* (2003), pp. 167–176.

[RBC98] ROSE C., BODENHEIMER B., COHEN M. F.: Verbs and adverbs: Multidimensional motion interpolation using radial basis functions. *IEEE Computer Graphics and Applications 18*, 5 (Sept. 1998), 32–40.

[SIC01] SLOAN P.-P. J., III C. F. R., COHEN M. F.: Shape by example. In *Proc. I3D'01* (2001).

[SM00] SHI J., MALIK J.: Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8 (2000).

[TP91] TURK M., PENTLAND A.: Eigen faces for recognition. *Journal of Cognitive Neuroscience 3*, 1 (1991).

[WG03] WILHELMS J., GELDER A. V.: Combining vision and computer graphics for video motion capture. *The Visual Computer 19*, 6 (Oct 2003).

$$\therefore$$

5.2   MOTION COMPRESSION USING PRINCIPAL GEODESICS ANALYSIS

M. Tournier, X. Wu, N. Courty , E. Arnaud, L. Reveret Computer Graphics Forum, Proceedings of Eurographics'09, Münich, Germany, april 2009.

Motion Compression using Principal Geodesics Analysis

M. Tournier[1], X. Wu[1], N. Courty[2], E. Arnaud[1] and L.Revéret[1]

[1]Laboratoire Jean Kuntzmann, INRIA, France
[2]Université de Bretagne-Sud

Abstract

Due to the growing need for large quantities of human animation data in the entertainment industry, it has become a necessity to compress motion capture sequences in order to ease their storage and transmission. We present a novel, lossy compression method for human motion data that exploits both temporal and spatial coherence. Given one motion, we first approximate the poses manifold using Principal Geodesics Analysis (PGA) in the configuration space of the skeleton. We then search this approximate manifold for poses matching end-effectors constraints using an iterative minimization algorithm that allows for real-time, data-driven inverse kinematics. The compression is achieved by only storing the approximate manifold parametrization along with the end-effectors and root joint trajectories, also compressed, in the output data. We recover poses using the IK algorithm given the end-effectors trajectories. Our experimental results show that considerable compression rates can be obtained using our method, with few reconstruction and perceptual errors.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism–Animation—Vision and Scene Understanding [I.2.10]: Motion—:

1. Introduction

Motion capture has became a ubiquitous technique in any domain that requires high quality, accurate human motion data. With the outcome of massively multiplayer online games, the transmission of huge quantities of such data can become problematic. Furthermore, in order to provide the user with a wide range of animations, several motion capture sequences often have to be used at once: motion data is either picked from a motion database, or constructed using blending or learning on existing data. In that case, a compact and easily editable representation of motion data could drastically improve user experience while decreasing the storage needs.

Raw motion capture data are indeed very large: they consist in the aggregation of sampled markers trajectories or joints orientations across time. With a sampling rate of 120 Hz and about 40 markers for the skeleton, the data size rapidly grows large. However, human motion exhibits inherent redundancies that can be exploited for compression purposes:

- **Temporal coherence**, thanks to which the motion can be keyframed with few loss of information,
- **Joints motion correlations**, which allows the representation of the motion in a smaller subspace.

We propose a novel, lossy compression of human motion data that exploits both of them. We first approximate the poses manifolds using Principal Geodesics Analysis (PGA) in the configuration space of the skeleton. This model yields a reduced, data-driven pose parametrization that is then used in an Inverse Kinematics (IK) algorithm to recover poses given end-joints positions only. Thanks to this algorithm, we are able to reconstruct the motion given only end-joints trajectories and the root joint's positions and orientations. The key idea is thus to perform compression by only storing this compact model along with the end-joints and root-joints trajectories. As we constraint end-joints positions during the decompression, typical compression artifacts such as foot skating are automatically reduced. By modifying these end-joints trajectories, one can also easily edit motions compressed using our method. Our experimental results show that significant compression rates with few distortion can be obtained using this approach, while keeping the possi-

bility to easily and naturally edit synthesized motion. Our method is easy to implement and runs quite fast on modern machines.

The rest of the paper is organized as follows: the related work is reviewed in section 2. We present an overview of the compression technique in section 3, followed by a brief presentation of the non-linear tools and their use in our system. Section 4 is dedicated to experimental results in which compression performances are evaluated. We conclude in section 5 by a discussion of the proposed method and possible future work.

2. Background and Related Work

2.1. Motion Compression

Though recent works on motion capture data compression can be found, the problem of motion compression has been mainly focused on animated meshes compression so far: those high-dimensional data often present high spatial and temporal coherence that can be exploited to reduce the data size. [Len99] detects parts of the mesh with rigid motion to encode only the transformation and the residuals. Correlations that may exist in parts of the moving object have also been exploited through the use of Principal Component Analysis (PCA) [SSK05] to compress the mesh vertices.

Skeleton motion also exhibits those cross-bones correlations. These are mainly exploited in optimization frameworks as they allow for search space dimension reduction. [SHP04] apply PCA on a group of similar motions in order to synthesize motion in such a reduced space. [GMHP04] use a probabilistic latent variable space to perform inverse kinematics that preserves stylistic properties. [LM06] detect motion segments in which joints positions lie in a reduced linear subspace and use PCA to reduce dimensionality for compression purposes.

Motion capture data also exhibits temporal coherence which can also be exploited to acheive compression: [LM06] use spline keyframing to compress the PCA projections of markers in motion segments. [Ari06] uses splines to represent global markers trajectories. The control points for a whole motion database are then compressed using clustered PCA. In both cases, working with global marker positions requires an additional pass of optimization to keep the bone length constant along the synthesized motions. Other methods use rotational data: [BPP07] adapt standard wavelet compression on joint angles by automatically selecting the basis elements in a way that minimizes quadratic error. However, high compression ratios can result in strange reconstructed paths due to the use of Euler angles.

Any lossy motion capture compression method, being orientations-based or positions-based, introduces errors that are likely to introduce various perceptual artifacts. The most striking is probably *foot skating*, which greatly penalizes the visual quality of synthesized motions. This artifact can be corrected using inverse kinematics(IK) techniques. However, using *style-based* IK [GMHP04] is often needed in order to correct the motion while preserving its visual identity.

2.2. Motion Metric

As with any lossy compression system, a central problem with motion capture data compression is the error metric used to evaluate the quality of the results. The problem in our case is that the metric should take perceptual features into account, which is a difficult task. While it is commonly accepted that the standard L^2 norm over markers positions is a weak indicator of the perceptual closeness of two animations, few works propose an alternative, efficient metric. [RP03] propose a study of user sensitivity to errors considering only ballistic motions. [RPE*05] try to evaluate the natural aspect of an animation. To do so, 3 classes of metrics are distinguished:

- Heuristic rules, that penalize the score of an animation when violated (*e.g.* physical laws)
- Perceptual metrics that highlight artifacts noticed by users (*e.g.* foot skating)
- Classifiers-based metrics trained on large data sets

The first two usually fail to quantify the natural aspect, or the style of an animation, but are good at detecting precise artifacts. The latter is based on the assumption that a human will perceive a motion as natural if it has already been seen a lot of times. On the contrary, an unusual motion will be perceived as unnatural. Such metrics often detect stylistic closeness successfully, but are highly dependent on the data set used for the training: they will fail for a natural motion that is not in the data set. Moreover, local physical anomalies or artifacts are often not detected. As a matter of fact, finding an accurate and robust metric for human motion perception remains, to the best of our knowledge, an open problem.

2.3. Non-linear Analysis

As mentioned earlier, two natural ways of compressing motion data are to exploit both temporal coherence and correlations in the motion of parts of the skeleton. To achieve this, one typically uses multiresolution and dimension reduction techniques. While well-known theoretical frameworks for these are available in the case of data lying in a linear space (such as wavelets, PCA), their extension to non-linear spaces (for instance, the space of rotations SO(3)) is not trivial and is a recent field of research.

[LS01] propose a *multiresolution* scheme for orientation data that allows editing, blending and stitching of motion clips. A potential application to compression is mentioned, though not developed. [RDS*05] generalize this scheme to symmetric Riemannian manifolds using exponential and logarithmic maps. The interpolating scheme used

can be seen as a special case of the so-called *lifting scheme* [Swe98]. The *lifting scheme* is an alternative way of defining wavelets and is presented in section 3.5. [RDS*05] propose an application to the compression of airplane headings with promising results.

The *dimension reduction* problem is often solved using descriptive statistical tools. Those tools typically yield a space that is more suitable for expressing the data: smaller dimension, orthogonal axis, most notably. The extension of known linear statistical tools to the non-linear case is not eased by the fact that many elementary results in the former case do not hold when dealing with more general spaces. For instance, the problem of finding the mean value of data lying on a sphere can no longer be expressed through probabilistic expected value, but has to resort to the minimization of geodesic distances [BF01]. Averaging rotations falls into this class of problem [Moa02].

Pennec [Pen06] gives basic tools for probabilities and statistics in the general framework of Riemannian manifolds. Fletcher [FLJ03], [FLPJ04] proposes a generalization of PCA to certain non-linear manifolds named Principal Geodesic Analysis (PGA), which consists in finding *geodesics* that maximize projected variance. He also presents an approximation of the analysis that boils down to a standard PCA in the tangent space at the mean of the data. It is presented in more details in section 3.3. An algorithm performing *exact* PGA for rotations is presented in [SCLS07], which shows that the number of principal geodesics needed for an exact reconstruction is not *a priori* bounded.

3. Proposed Method

3.1. Motivations - Overview

In this section we give an overview of our motion capture data compression method. Most approaches to human motion compression exploit global markers positions to achieve compression. While this has some advantages, such as speed and the use of well-known frameworks, the biggest drawback is that the constant bone-length of the skeleton cannot easily be guaranteed, which can introduce undesired limbs deformations. A post-processing pass is needed for this constraint to be enforced. Yet, this additional process can itself introduce artifacts. We want to address this problem by working on orientations rather than positions. However, because of the hierarchical nature of the skeleton, even slight errors in reconstructed orientations can lead to significant positions errors for end-joints. The most notable artifact of this kind is probably foot skating, which greatly penalizes the perceptual quality of synthesized animations. We intend to work around this by building a pose model from the animation clip: this model will allow us to synthesize poses that match given end-joints constraints, while staying close to the input data.

A pose is defined as a vector of rotations that describe the orientations of the skeleton's joints. It is therefore an element of $SO(3)^n$, where $n \in \mathbb{N}$ is the number of joints in the skeleton. Given a motion composed of $m \in \mathbb{N}$ poses, we use the *Principal Geodesics Analysis* to build a descriptive model of those pose data, keeping only the leading principal geodesics. This model is then used in an inverse kinematics system to synthesize poses that both match end-joints constraints *and* are close to the input data. Given this pose model, we only have to store the compressed end-joints trajectories as well as the root joint's positions and orientations (also compressed) in order to recover the motion using IK. The compression/decompression pipeline is presented on figure 1.
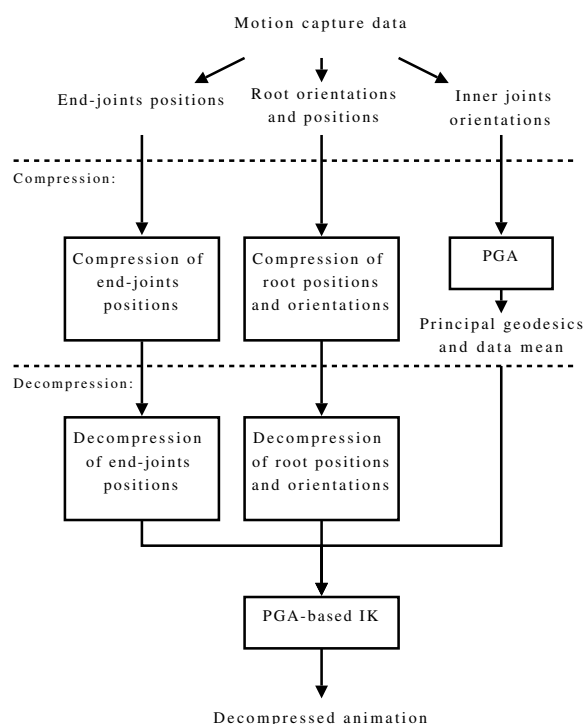
Figure 1: *Flow diagram for the compression pipeline*

We now briefly present the non-linear tools employed throughout this paper, as well as their use in our algorithm.

3.2. Lie Groups - Exponential Map

The space for three-dimensional rotations is a particular case of a Lie group. A Lie group is a group which is also a differentiable manifold, and for which the inverse and the group operations are differentiable. Such manifolds have been extensively studied and used in robotics to describe configuration spaces of articulated bodies [MSZ94]. We give a quick definition of the *exponential map* used throughout this paper. For a complete theoretical treatment, see [JJD00]. Let *G* be

a Lie group. The *exponential* map is a mapping from the tangent space of G at the identity (that is, the Lie algebra of G, $\mathfrak{g}$) to the group itself G. For every tangent vector of the Lie algebra $v \in \mathfrak{g}$, one can define a left-invariant vector field v^L by left-translation of v. Let $\gamma_v(t)$ be the unique maximal integral curve of such a vector field, the exponential map is then defined by $\exp(v) = \gamma_v(1)$. It is the unique one-parameter subgroup of G with initial tangent vector v. It is a diffeomorphism in a neighborhood of $0 \in \mathfrak{g}$, and the inverse mapping is called the *logarithm*. In the case of a Lie group endowed with a compatible Riemannian structure, such as SO(3)), the Riemannian and Lie exponential coincide. This means that we can compute geodesic curves (*i.e.* locally length-minimizing curves) as well as measuring lengths of such curves using the Lie exponential. The length of the shortest geodesic curve(s) between two points is called the *geodesic distance*.

For matrix Lie groups (*i.e.* subgroups of $GL_n(\mathbb{R})$), the exponential is defined by the usual exponential power series: $\exp(M) = \sum_{i \geqslant 0} \frac{M^i}{i!}$. For rotations, the sum of this series is known as the *Rodrigues'* formula [Gra98]. One can define an exponential mapping at any point $g \in G$ of the group simply by translating (in the group's operation sense) the standard exponential map: $exp_g(v) = g.exp(v)$.

In the case of data lying in an abstract manifold such as SO(3), one can no longer define the mean as a weighted sum of elements. Instead, the *intrinsic* mean is defined as a point that minimizes the geodesic distance with respect to all the points considered. It can be computed by an optimization algorithm (see [FLJ03] or [Pen06]) which uses the exponential map and that usually converges in a few iterations.

3.3. Principal Geodesics Analysis

The Principal Geodesics Analysis is an extension of the Principal Component Analysis introduced by Fletcher in [FLJ03], [FLPJ04]. Its goal is to describe variability in Lie groups that can be given a Riemannian structure compatible with the algebraic one. Such groups include $(\mathbb{R}^n, +), (\mathbb{R}, \times), (SO(3), \circ)$ as well as any direct product between them. The idea behind PGA is to project data onto *geodesics* in a way that maximizes the projected variance. In the linear case, the geodesics are simply lines between two points, and the PGA then boils down to standard PCA. However, the projection onto a geodesic curve cannot be defined analytically in the general case, and therefore involves a minimization algorithm. In order to avoid this, Fletcher proposes to approximate the projection onto geodesics by a linear projection in the tangent space at the intrinsic mean of the data, using the exponential mapping at that point. Under this approximation, the principal geodesics' directions can be computed by a standard PCA of the linearized data. The fact that the linearization is done at the data mean conveniently guarantees its coordinate-invariance, but also ensures that the induced distortion is minimized.

We employed the PGA framework to describe the variability of the *inner* joints orientations during a motion. However, since the *exact* PGA computation in $SO(3)^n$ remains (to our best knowledge) an open problem, we used the approximation proposed by Fletcher to compute the principal geodesics' directions. These geodesics can be looked upon as the *eigenposes* of the skeleton during the sequence. We did not include the root joint's orientation in the analysis: indeed, it is only poorly correlated with the pose of the skeleton, and using it in the PGA can alter the resulting principal geodesics. As mentioned earlier, the pose of the skeleton is represented by a vector of the direct product $SO(3)^n$, where n is the number of joints of the skeleton. Applying the *approximate PGA* to the poses data from a motion with $m \in \mathbb{N}$ frames yields:

- The intrinsic mean of the data, $\mu \in SO(3)^n$
- $k \in \mathbb{N}$ tangent directions $(v_j)_{1 \leqslant j \leqslant k}$, where each $v_j \in \mathfrak{so}(3)^n \simeq \mathbb{R}^{3n}$ uniquely defines a geodesic of $SO(3)^n$
- A set of coordinates $T = (t_{i,j})$ where $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant k$, where the ith row is the approximate projection of the ith pose over the k geodesics

The ith pose can then be recovered partly using the k leading geodesics with:

$$p_i = \mu . \prod_{j=1}^{j=k} e^{t_{i,j} . v_j}$$

This parametrizes the approximate poses manifold using the *canonical coordinates of the second kind* (ccsk, see [MSZ94]). One can think of the reconstruction formula as a weighted composition of the k first eigenposes. Note that here the exponential over the direct product $\mathfrak{so}(3)^n$ is used. Examples of principal geodesics extracted from motion capture data can be seen on figure 2.

As shown on figure 3, the number of principal geodesics needed to represent 99% of the input data variance is generally inferior to 20. For motions with stronger correlations, such as walking motions, 10 geodesics are in most cases enough to express 95% of the input variance.

By only considering the $k \leqslant n$ first modes of the PGA, we obtain a new reduced parametrization of a motion in terms of geodesics coordinates. We show next how such a reduced pose parametrization can be used to perform inverse kinematics.

3.4. PGA-based Inverse Kinematics

The reduced parametrization with geodesics coordinates allows us to define a function $f : \mathbb{R}^k \to \mathbb{R}^{3d}$ that maps a set of geodesics coordinates $x \in \mathbb{R}^k$ to the global space positions of $d \in \mathbb{N}$ end-effectors: $y \in \mathbb{R}^{3d}$. This function is the composition of the reduced pose parametrization by the ccsk $h : \mathbb{R}^k \to SO(3)^n$ and the classical direct kinematics function, which maps a skeleton pose to the global position of
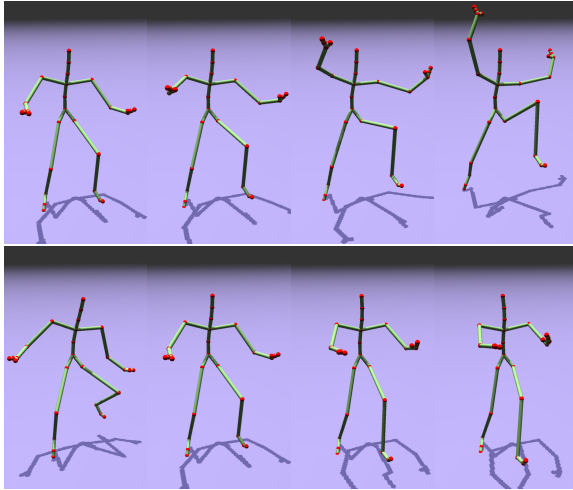
Figure 2: *Two examples of principal geodesics extracted from one* break dance *motion using PGA. Here, the corresponding poses are given for 4 different geodesic coordinates. The top left picture shows the mean pose (with a zero geodesic coefficient).*

the d end-effectors, $g : SO(3)^n \to \mathbb{R}^{3d}$. The derivative of g at the pose $x \in SO(3)^n$ simply maps instant rotation vectors for each joint to the linear velocities of the end-effectors.

Since $h(x) = \mu . \prod_{j=1}^{j=k} e^{x_j . v_j}$ is a product of differentiable functions (the exponentials) in a Lie group, h is therefore differentiable. Each partial derivative of h with respect to x_j can be easily computed due to the ccsk parametrization using the adjoint Ad map over $SO(3)$ (see for instance [MSZ94], p.116). We are eventually able to compute the whole Jacobian matrix J_f of the function f using chain rule. We then use this Jacobian in a least square optimization method, such as the well-known Levenberg-Marquardt algorithm, in order to find the geodesic coordinates $x_j^\star$ that best match the given end-effectors constraints $y_0 \in \mathbb{R}^{3d}$:

$$x^\star = \underset{x \in \mathbb{R}^k}{\mathrm{argmin}}(\|f(x) - y_0\|^2)$$

The benefits of using this method are threefold:

- The optimization is done in a **much smaller space** than traditional IK (usually 30 degrees of freedom): this not only speeds up the process, but also better constraints the IK problem.
- The geodesics being principal poses modes, **the optimization naturally exploits correlations** between joints to reach the objectives, resulting in a more natural pose.
- The geodesics formulation allows a quick computation of the Jacobian J_f used in the optimization, thus eliminating the need for numerical differentiation.

The main drawback is that the geodesics yield a limited
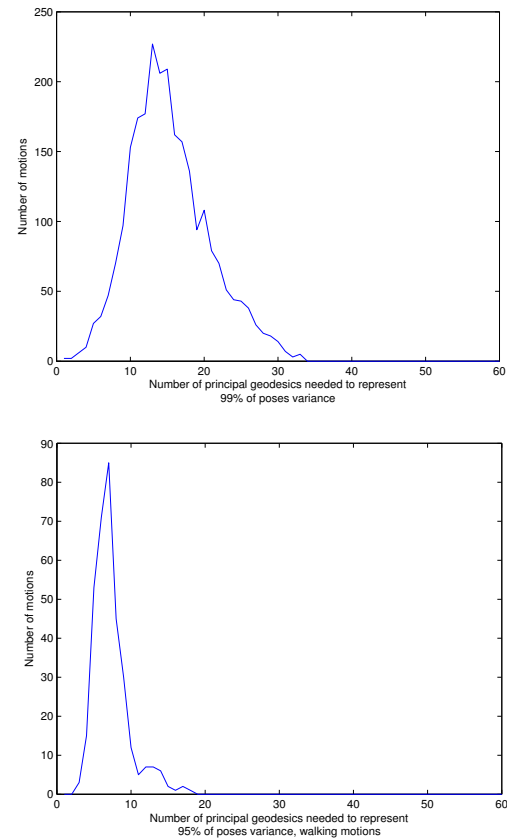
Figure 3: *Histograms showing the numbers of geodesics needed to achieve given variance reconstruction, using approximate PGA. Top: 99% of variance for the whole CMU database. Bottom: 95% of variance for only walking motions.*

reachable space: our IK works better in a neighborhood of the input data. However, since we are only interested in recovering poses belonging to the input data, this is not really a problem. Of course, the more geodesics we use, the larger the reachable space. In our experiments, 10-12 modes are generally sufficient to perform IK on 4 or 5 IK handles at once. Apart from compression, this IK system can also be used independently in real-time as seen on figure 4 for interactive motion editing.

In order to recover an animation sequence using the PGA-based IK, one needs the following data:

- The inner orientations mean and the k leading principal geodesics
- The end-joints trajectories across time
- The root joint's orientation and position across time

To recover each frame, we first express the end-joints trajectories in the root joint's frame, then perform IK as presented
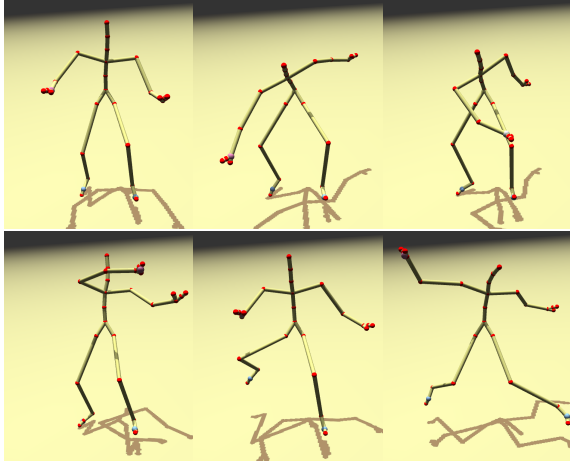
Figure 4: *The PGA-based IK system used in real-time. The input motion is a* break dance *sequence from the CMU database. There are three IK handles in this example that can be manipulated by the user: one on each foot and one on the right hand. Here the optimization is done using 10 geodesics.*

earlier. Doing so already results in a good compression of the data since we only have to store 7 trajectories (2 for the root and 5 for the end-joints) instead of more than 30 found in the original motion. Of course, since those trajectories present a high degree of temporal coherence, we will exploit it to further compress the motion data.

3.5. Multiscale Representation for Orientation Data

In order to compress the root's orientation, we use the multiscale representation for manifold data introduced in [RDS*05]. As a particular case of the so-called *lifting scheme* [Swe98], it can be summed up as follows. Let D be the set of data we want to represent in multiscale:

- D is first **partitioned** into two parts, A and B
- Data in A are then used to **predict** the data in B, using some prediction operator
- The **differences** between the *predictions B_A* and the actual B data form the *details*

The process is then iterated over the set A until there are no data remaining. By doing so, one creates a collection of decreasing-size levels of details. This "pyramid" is the multiscale representation of the original data. For this representation to be useful in compression, one generally wishes to partition the data so that the prediction step is as accurate as possible. In that case, the details needed to correct the prediction are small and can hence be omitted with few errors.

In the case of time-dependent orientation data, D can be represented by a collection of rotations $d = (d_i)_{1 \leqslant i \leqslant m}$,

where m is the number of samples. In order to exploit temporal coherence, we simply sub-sample the data by a factor 2 to partition the data. The prediction step is done using the tangent spline interpolation described in [RDS*05]. To interpolate between two rotations r_0 and r_1, we express r_{-1}, r_0, r_1, r_2 in $T_{r_0}(\mathrm{SO}(3))$ using the logarithm map. We then interpolate those tangent vectors *in the tangent space* using splines at $t = \frac{1}{2}$, and finally go back to $\mathrm{SO}(3)$ with the exponential map. The difference between the predicted value $\widetilde{r}_{\frac{1}{2}}$ and the original data r_{orig} is stored as $d = \log(\widetilde{r}_{\frac{1}{2}}^{-1}.r_{\mathrm{orig}})$. The original data can eventually be recovered using $r_{\mathrm{orig}} = \widetilde{r}_{\frac{1}{2}}.\exp(d)$.

We could have used simple SLERP [Sho85] prediction between r_0 and r_1 (as in [LS01]). This corresponds to simple linear approximation in the tangent space at one of the two points. However, this lead to a piecewise SLERP reconstructed signal when omitting levels-of-details, which presents discontinuities of the first derivatives that penalize the visual quality of the result. Instead, the use of tangent spline interpolation results in a smooth reconstructed signal even in the case of missing data.

3.6. Putting Everything Together

After the principal geodesics have been extracted from the input motion using approximate PGA, global end-joints trajectories can be compressed using any linear compression method. The root orientation is eventually compressed using the multiscale representation presented in section 3.5. For the sake of consistency, our implementation uses the presented multiscale scheme for both orientations and positions, but any suitable temporal coherence-based compression technique could work. The decompression phase consists in decompressing the global trajectories as well as the global root orientation, then expressing the end-joints positions in the root joint's frame, and eventually performing PGA-based IK to recover poses.

Let us now give an estimation of the data size needed to store an animation using our technique. Each of the k geodesics kept after the PGA is a vector of $\mathbb{R}^{3n}$, which is roughly the size of one motion frame. The mean value of the inner joints can also be stored as a vector of $\mathbb{R}^{3n}$ using the exponential map. The data needed for the PGA reconstruction can hence be stored in a $s_{\mathrm{PGA}} = (k+1) \times 3n$ matrix. The global root orientations and positions as well as the 5 end-joints' positions can easily be compressed by a factor $8 = 2^3$ by omitting 3 levels of details: each time we get rid of one level, we divide the data size by two. All those trajectories together can be encoded in a $s_{\mathrm{traj}} = 3(2+5) \times \frac{m}{8}$ matrix. Given an initial animation with size:

$$s_{\mathrm{orig}} = m \times 3(n+1) = O(m \times n)$$

whereas the compressed version using our algorithm, keeping k geodesics, will have the size:

$$s_{\mathrm{compressed}} = s_{\mathrm{PGA}} + s_{\mathrm{traj}} = O(m+n)$$

Examples of compression ratios obtained using our technique can be found in section 4.

As it involves an optimization process for decompressing each frame, our method is subject to the classic pitfalls of those algorithms: degenerate Jacobian, local extrema, smoothness of the reconstructed animation. In our experiments, we found that the first happen almost only when the constraints are out of the reachable space, and such cases do not happen when reconstructing the original signal. The smoothness of the solution can be enforced by adding a penalty term in the optimization so that the solution for the current frame decompression is searched in a neighborhood of the previous one:

$$x^\star = \operatorname*{argmin}_{x \in \mathbb{R}^k}(\|f(x) - y_0\|^2 + \lambda\|x - x_{\text{previous}}\|^2)$$

where λ is a given user threshold. This inevitably leads to drifting problems due to the existence of local extrema. In our experiments however, setting $\lambda = 1$ always gave correct results.

Before presenting the results obtained with our method, let us explicit what we feel are the main benefits of an approximate PGA over a PCA of the standard exponential maps for poses parametrization, since the two techniques might seem similar. First of all, using Fletcher's approximation at the intrinsic mean leads to a coordinate-invariant and distortion-minimized linearization of the data. This property improves the quality of the statistical analysis since it only depends on the input motion, and not on its *parametrization* (*e.g.* the choice of the reference pose). Secondly, the pose parametrization along the principal geodesics is easily differentiated, which is highly desirable for any pose-related optimization, since no finite differences approximations need to be computed.

4. Experimental Results

We present here the compression rates of our algorithm on selected motions from the Carnegie Mellon University's Graphic Lab motion capture database available at http://mocap.cs.cmu.edu. We chose motions with different characteristics of length, diversity, and dynamics as shown on table 1.

ID	Subject/Trial	Description	#Frames
1	09/06	Running, short	141
2	17/08	Walking, slow	6179
3	15/04	Various, dancing, boxing	22948
4	85/12	Breakdance	4499
5	17/10	Boxing	2783

Table 1: *Motion capture clips used in our experiments*

As stated in section 2.2, no really robust and efficient metric is available to assess the quality of the reconstructed animations. However, for the sake of results comparisons, we used a distortion rate as the one defined in [KG04] to evaluate the quality of the reconstruction. This distortion rate is defined as:

$$d = 100\frac{\|A - \tilde{A}\|}{\|A - E(A)\|}$$

where A is the $m \times 3n$ matrix containing absolute markers' positions at each frame for the original motion, $\tilde{A}$ is the same matrix for the decompressed animation, and each row of $E(A)$ contains the mean markers' positions with respect to time.

Table 2 shows the obtained compression ratios and distortion rates for different combinations of geodesics numbers and trajectories levels of details. Table 3 shows the results obtained by [LM06], who holds the best compression rates at the time of writing. Note that we always used 5 end-joints in our tests, but more could be used if a higher quality is required. As expected, our method works best when the spatial and temporal coherence is high: a rather slow walking motion can be compressed 185 times with few reconstruction errors, whereas a highly dynamic breakdance sequence is only compressed 118 times for about the same distortion rate. This table shows that our technique allows substantial compression rates improvement over existing techniques, with very limited distortion. The accompanying video additionally shows that most of time, the differences between compressed and original animation are hard to find out, unless the two motions are displayed at the same position.

ID	1	2	3	4	5
#Geodesics	6	12	17	15	12
#LOD (root)	4/9	8/14	12/16	9/14	8/13
#LOD (end-joints)	4/9	8/14	12/16	9/14	9/13
Compression ratio	1:18	1:182	1:69	1:97	1:61
Distortion rate (%)	0.36	0.049	1.55	0.56	0.49
Decompression time (msec/frame)	7.88	16.2	30.6	20.42	15.97

Table 2: *Compression rates, distortion errors and decompression times for the selected motions using our technique. Different combinations of geodesics numbers and trajectories level of details are presented*

When too few geodesics are used in the IK, the reachable pose space gets too small and the synthesized poses sometimes fail to match all the given constraints. On the contrary, once enough geodesics have been selected, further increases on that number only result in a slower optimization time. In our experiments, 10 to 15 geodesics are sufficient in most cases to yield a large enough pose space. For long motions

Sequence	Compression ratio	Distortion rate	Decompr. time (msec/frame)
Jumping, bending, squats	1:55.2	5.1	0.7
Long breakdance sequence	1:18.4	7.1	0.7
Walk, stretches, punches, drinking	1:61.7	5.1	0.7
Walk, stretches, punches, kicking	1:56.0	5.4	0.7

Table 3: *Compression rates presented in [LM06] using PCA on motion segments. We achieve substantial compression rates improvement with fewer distortion, though our decompression pass takes longer.*

in which clear distinct motion behaviors occur, a segment-based approach similar to [LM06] may be used. This could improve the accuracy of each different pose model, resulting in more natural poses for each behaviors, and possibly better conditioning the Jacobian matrix used in the optimization. The transition between different models would have to remain smooth however, which is not an easy task. As expected, when the compression for the end-joints trajectories is set too high, artifacts start to appear as the feet contacts on the ground are smoothed too much. In the same way, too high compression over the root joints' position and orientations causes the skeleton to slide, as hung in the air. The acceptable compression ratio for those trajectories highly depends on the dynamics found in the animations. In any case, quantization may be used to compress the trajectories reconstruction errors while controlling additional overhead.

Since we are performing a statistical analysis to represent the motion's poses, strong outlier poses will be more difficult, if not impossible, to reconstruct correctly. In practice, such a situation could arise if the character holds a very specific pose for a very short amount of time with respect to the duration of the motion. We did not encounter such cases in our experiments. The compression time depends on the length of the animation, as it only involves the intrinsic mean of pose data calculation, and a PCA of the linearized rotations in the tangent space at that point. In practice, it is very inferior to the decompression time, during which an optimization is performed for each frame to reconstruct poses given end-joints constraints. Our implementation was realized in C++ on a Dell 390 workstation, with dual 2.6 Ghz CPU and 4 GB memory. Due to the nature of the calculations involved in the optimization pass, we believe that a GPU implementation might be engineered by sampling the research space.

5. Conclusion and Future Works

We present a novel method for human motion capture data compression exploiting both temporal and spatial coherence to achieve high compression ratios with few perceptual distortion. Our experiments show that the use of a compact pose model allows to successfully recover poses given only end-joints positions. As the end-joints and root joint's trajectories present high temporal coherence, they can also be compressed in order to further improve compression rates. A particularly appealing aspect of our technique is that the pose model may also be used for editing compressed motions by employing the very same algorithm.

Though the inverse kinematics algorithm we presented is able to run in real-time on a modern machine, the decompression times are still longer than for other motion capture data compression techniques. However, our implementation could still be improved. The compression technique used for end-joints trajectories could also be enhanced to better reconstruct sharp features, such as foot contacts. The use of a suitable wavelet compression could lead to better results. We also did not exploit the linear correlations present in the end-joint's positions: applying a compression technique similar to the one presented in [LM06] to these markers could even improve compression performances, either allowing to further reduce data size, or to increase the number of constraint joints in the IK. If more quality is required, quantization could also be employed to improve the reconstruction of joint's trajectories by compressing the errors with controllable size overhead.

Finally, our data-driven, PGA-based IK algorithm is very promising as it allows to perform *data-driven* inverse kinematics without resorting to computationally expensive learning, such as the one presented in [GMHP04]. Thus, it could find application in multiple areas such as recovering poses and motion from videos. The run-time optimization also seems to be faster, which could make it suitable for real-time applications.

Acknowledgements
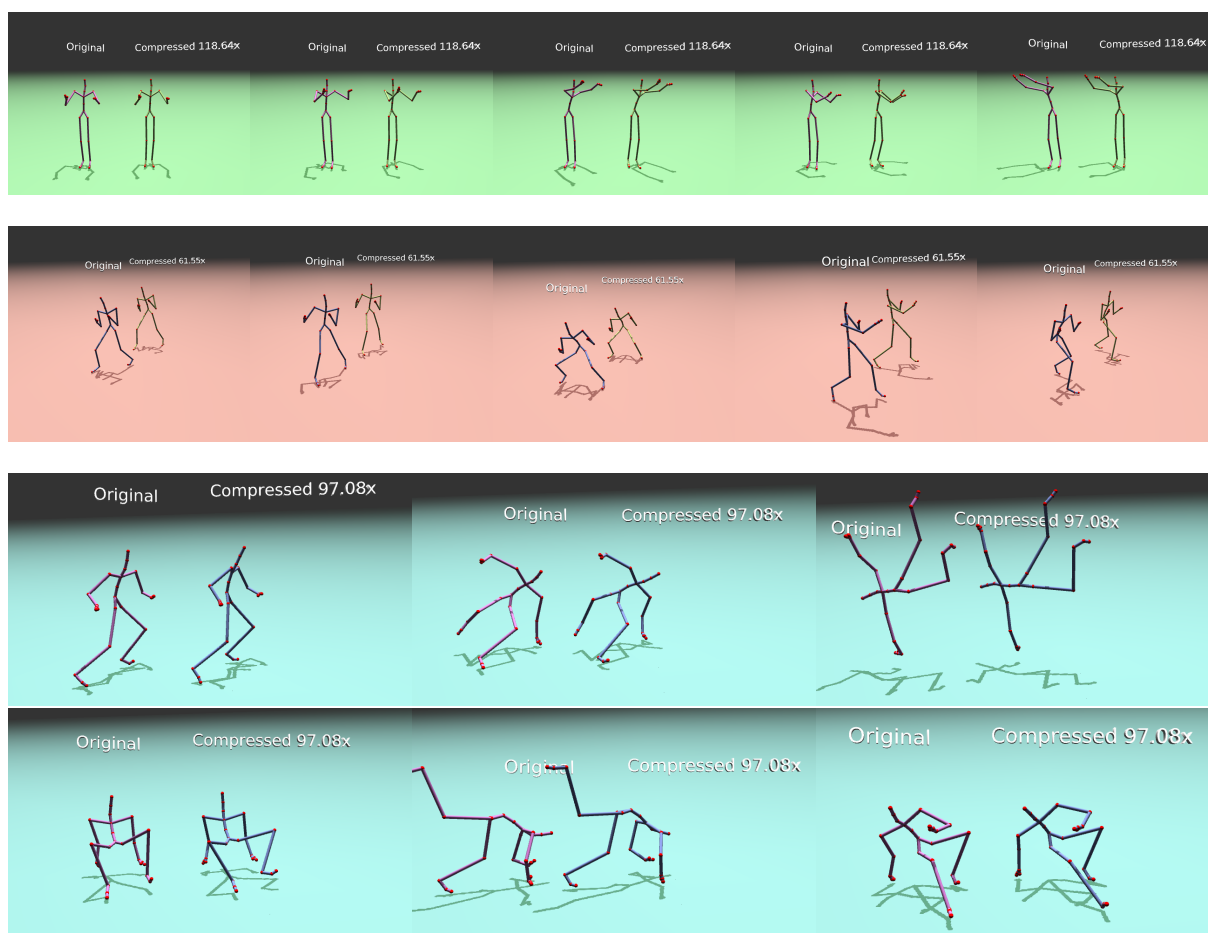
Figure 5: *3 animations from the CMU database compressed using our technique*

References

[Ari06] ARIKAN O.: Compression of motion capture databases. *ACM Trans. Graph. 25*, 3 (2006), 890–897.

[BF01] BUSS S. R., FILLMORE J. P.: Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics 20*, 2 (2001), 95.

[BPP07] BEAUDOIN P., POULIN P., PANNE M. V. D.: Adapting wavelet compression to human motion capture clips. In *GI '07: Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), ACM, pp. 313–318.

[FLJ03] FLETCHER P. T., LU C., JOSHI S. C.: Statistics of shape via principal geodesic analysis on lie groups. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2003 Proceedings CVPR-03* (2003), pp. I–95.

[FLPJ04] FLETCHER P. T., LU C., PIZER S. M., JOSHI S. C.: Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging 23*, 8 (2004), 995.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIC Z.: Style-based inverse kinematics. In *ACM SIGGRAPH 2004 Papers on - SIGGRAPH 04 SIGGRAPH 04* (New York, NY, USA, 2004), ACM Press, p. 522.

[Gra98] GRASSIA F. S.: Practical parameterization of rotations using the exponential map. *journal of graphics tools 3*, 3 (1998), 29–48.

[JJD00] J. J. DUISTERMAAT J. A. C. K.: *Lie Groups*. Springer, 2000.

[KG04] KARNI Z., GOTSMAN C.: Compression of soft-body animation sequences, 2004.

[Len99] LENGYEL J. E.: Compression of time-dependent geometry. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (New York, NY, USA, 1999), ACM, pp. 89–95.

[LM06] LIU G., MCMILLAN L.: Segment-based human motion compression. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 127–135.

[LS01] LEE J., SHIN S. Y.: A coordinate-invariant approach to multiresolution motion analysis. *Graphical Models 63*, 2 (2001), 87.

[Moa02] MOAKHER M.: Means and averaging in the group of rotations. *SIAM J. Matrix Anal. Appl. 24*, 1 (2002), 1–16.

[MSZ94] MURRAY R. M., SASTRY S. S., ZEXIANG L.: *A Math-*

ematical Introduction to Robotic Manipulation. CRC Press, Inc., Boca Raton, FL, USA, 1994.

[Pen06] PENNEC X.: Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision 25*, 1 (2006), 127.

[RDS*05] RAHMAN I. U., DRORI I., STODDEN V. C., DONOHO D. L., SCHRODER P.: Multiscale representations for manifold-valued data. *Multiscale Modeling & Simulation 4*, 4 (2005), 1201–1232.

[RP03] REITSMA P. S. A., POLLARD N. S.: Perceptual metrics for character animation. In *ACM SIGGRAPH 2003 Papers on - SIGGRAPH 03 SIGGRAPH 03* (New York, NY, USA, 2003), ACM Press, p. 537.

[RPE*05] REN L., PATRICK A., EFROS A. A., HODGINS J. K., REHG J. M.: A data-driven approach to quantifying natural human motion. In *ACM SIGGRAPH 2005 Papers on - SIGGRAPH 05 SIGGRAPH 05* (New York, NY, USA, 2005), ACM Press, p. 1090.

[SCLS07] SAID S., COURTY N., LEBIHAN N., SANGWINE S.: Exact principal geodesic analysis for data on so(3). In *Proc. of the 15th European Signal Processing Conference (EUSIPCO 2007)* (Poznan, Poland, September 2007).

[Sho85] SHOEMAKE K.: Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM, pp. 245–254.

[SHP04] SAFONOVA A., HODGINS J. K., POLLARD N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM SIGGRAPH 2004 Papers on - SIGGRAPH 04 SIGGRAPH 04* (New York, NY, USA, 2004), ACM Press, p. 514.

[SSK05] SATTLER M., SARLETTE R., KLEIN R.: Simple and efficient compression of animation sequences. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA 05 SCA 05* (New York, NY, USA, 2005), ACM Press, p. 209.

[Swe98] SWELDENS W.: The lifting scheme: a construction of second generation wavelets. *SIAM J. Math. Anal. 29*, 2 (1998), 511–546.

$\therefore$

5.3 MODAL LOCOMOTION : ANIMATING VIRTUAL CHARACTERS WITH NATURAL VIBRATIONS

Modal Locomotion:
Animating Virtual Characters with Natural Vibrations

P. G. Kry[1] and L. Reveret[2] and F. Faure[2] and M.-P. Cani[2]

[1]McGill University, School of Computer Science
[2]Grenoble Universities & CNRS, Jean Kuntzmann Lab, INRIA Grenoble Rône-Alpes

Abstract

We present a general method to intuitively create a wide range of locomotion controllers for 3D legged characters. The key of our approach is the assumption that efficient locomotion can exploit the natural vibration modes of the body, where these modes are related to morphological parameters such as the shape, size, mass, and joint stiffness. The vibration modes are computed for a mechanical model of any 3D character with rigid bones, elastic joints, and additional constraints as desired. A small number of vibration modes can be selected with respect to their relevance to locomotion patterns and combined into a compact controller driven by very few parameters. We show that these controllers can be used in dynamic simulations of simple creatures, and for kinematic animations of more complex creatures of a variety of shapes and sizes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Animation

1. Introduction

Locomotion is a fundamental activity of living creatures. It is possible that animals have evolved to optimize this energetically expensive function, but they also learn to use their body in efficient ways to produce locomotion styles that depend on speed and task [Ale96]. For instance, an animal might use a slow gait to search for shelter, and use a different faster gait to escape from a predator. Because the different gaits of a particular animal strongly depend on the morphology of that animal, it is possible that we can deduce information about plausible modes of locomotion just by observing that animal's musculoskeletal structure.

We present a method for generating animation of locomotion, which does not use motion capture, but instead uses only information of the physical model of the virtual animal. For a given model we compute a set of basis motions that can be combined in small number to describe different gaits for that model. Changes in the virtual animal model, such as weight, injury, rest pose, and muscle tension and relaxation, all produce subtle yet important differences in the basis motions and the resulting generated animation.

Skeletal structure, deformation of tissues, and stiffness of muscles and tendons all play an important role in the dynamics of real animals. An appealing strategy for modelling the dynamics of a virtual animal is to simplify the biomechanical system as a set of rigid bones connected by compliant joints, where the stiffness of the joints approximates the combined action of tendons, the activation of muscles, and the deformation of the surrounding tissues. Even with this simplified model of the biomechanics, most vertebrates have hundreds of degrees of freedom making it challenging to animate virtual models of these animals. While it is possible to remove some degrees of freedom from the skeletal structure (for instance, a simplified spine), we instead perform a modal analysis to identify a reduced set of coordinated joint motions that are useful for animating a virtual animal.

The natural vibrations identified by the modal analysis provide an analytical solution to the passive response of the elastic dynamics of the system [PW89]. We focus on lower frequency modes because at a given energy level they involve larger displacements than higher frequency modes and damping dissipates this energy more slowly. The intuition is that our virtual animals might take advantage of swinging

limbs and elastic compression at joints in a dynamic cycle that produces an efficient locomotion; efficient because it is the passive response to a given set of initial conditions, or more accurately, almost passive with only small regular pushes from the muscles being necessary to inject lost energy back into the system.

In this paper we show that modal vibrations can be combined in small number to produce locomotion controllers for a variety of virtual animals. The locomotion controllers can be dynamic, but they can also be purely kinematic. This is a significant contribution of our approach, since it is very easy to create kinematic motions with our technique which have a natural appearance due to the physically based nature of the basis motions. Instead of animating individual joints, for instance, the hip joints of a human, we can animate a mode that exhibits a leg swinging motion. But this leg swinging mode will also include nuanced motion in the rest of the body (such as upper body and spinal motions that conserve angular momentum). In general, searching for controller parameters is simplified since we only need to consider a small number of intuitive modes rather than a large number of degrees of freedom. This search can be easily done by hand, but in many cases it is also possible to do so automatically with a set of heuristics.

2. Related Work

Modeling and animating locomotion has attracted a lot of attention for many years in Computer Graphics, in addition to a huge amount of research in robotics and biomechanics. Locomotion controllers have been proposed for the computer animation of all sorts of virtual creatures. Motor control has been used to animate cockroaches [MZ90], fish [GT95], and birds [WP03]. Sun and Metaxas [SM01] demonstrated a gait generation technique which produces kinematic motion of human walking on a variety of different terrains. Also with respect to human models, Hodgins proposed controllers for human athletics [HWBO95] and later studied the way to adapt some of them to new characters [HP97]. This last work demonstrates that tuning a controller to a different morphology, even within the same species, is far from obvious. Therefore, there is interest in working towards more automatic generation of animation of arbitrary virtual animals [HRE*08].

Controlling locomotion is complex because many degrees of freedom must be managed simultaneously. The large dimensionality of the search space is a significant problem in designing a controller. A common approach for automatic generation of controllers is off-line optimization (maximization of the covered distance) over a large space of possible controllers. Sims used genetic programming to cope with the size of the search space, and evolved both the creature's structure and the controller simultaneously [Sim94]. In contrast, our interest is in the generation of plausible controllers for a given input bio-mechanical structure. In other work,

van de Panne [vF93, vKF94, vdP96] used optimization to tune the weights of simplified neural networks representing controllers. Grzeszczuk and Terzopoulos [GT95] introduced a multi-level learning process and a compact representation of controllers based on Fourier analysis, enabling them to be combined within a sensor feedback loop. They later developed neural networks as a way to efficiently emulate the control of physically based motion [GTH98].

The automation of controller generation can be simplified through *a priori* knowledge or example data to speed up or guide results. For instance, general momentum parametric templates inspired by biomechanics can be used to create complex dynamic motion [LP02]. This idea of motion-patterns is also illustrated in the work of [WP03] to represent the motor control of the beat of a bird wing. While recent work in graphics is now largely focusing on motion capture, much of this work is still concerned with optimal physically based modifications or identifying underlying physical parameters. Low-dimensional subspaces learned from motion capture databases can help with optimization of realistic human animation [SHP04]. Likewise, when an accurate database of real motion data is available, Liu et al. [LHP05] have shown that physical properties of the model (e.g., stiffness) can be estimated. Motion capture of quadrupeds, however, is not as widely available as it is for humans. However, foot prints can be successfully used as input data and combined with a simplified physically based representation of the skeleton to resythesize skeletal motion [TvdP98]. Other work has exploited side-view video data to reconstruct some types of 3D motion [Wil97, FRDC04]. In contrast, the only *a priori* knowledge we use is the morphological parameters (that is, the animal shape, size, mass, rest pose, joint stiffness). These parameters also effectively capture style (as in [LHP05]). Neff also demonstrates that modulating stiffness, specifically muscular tension and relaxation, is necessary for tuning the expression and style of an articulated motion [NF02]. This concept is also critical in our work because muscular tension (stiffness) directly affects the vibration modes and frequencies. Similar to our work, Faloutsos et al. use a reduced number of deformation modes, though user designed, to simplify the control problem for deformable models [FvdPT97].

The term passive dynamics was coined by McGeer [McG90] to describe an approach of controlling robotic movement that takes advantage of the passive movements, such as swinging legs and arms, rather than active control with motors applying torques at joints. Passive dynamics is relevant to locomotion because it is energetically efficient (that is, locomotion by passive dynamics is effectively free). Passive dynamic locomotion has been demonstrated with mechanical leg systems (no motors and no control) that produce dynamic walk cycles that resemble human walking [McG90, CWR01]. Collins et al. [CRTW05] provide a good recent survey of the vast amount of work done by a variety of people. This survey

also addresses work on efficient walkers (e.g., [Kuo02]) that walk on flat ground in contrast to the completely passive walkers which only walk down a gentle slope. In general, this view of efficient locomotion is a key inspiration to our work. Our model of locomotion as a superposition of natural vibration is also related to work on coupled oscillators in neural models for controlling human locomotion [Tag95]. The use of central pattern generators for modeling and controlling locomotion in robots and simulation has been and is still an active area or research; see [Ijs08] for a recent survey.

Finally, much like other work [MA90,RH91], we note that a skeletal system stores energy during locomotion through the compression of elastic joints. While the idea that this is a key component of locomotion is well known in biomechanics [Ale88], it is most commonly illustrated with simple models since these simple models are still a powerful tool for answering important questions about different gaits. A key contribution of our technique is that modal analysis similarly reduces the complexity of a 3D model while preserving important characteristics and behaviour of the original model, such as mass distribution and spinal flexion.

3. Dynamics of Virtual Animals

Our approach to creating virtual animal locomotion starts with the physically based model of the creature we want to animate. In practice, we use a slightly simplified skeleton to avoid slow simulations involving every vertebra of the spine (nonetheless, we could easily use the full model in our vibration analysis). For example, the dog model in Figure 1 has only 80 degrees of freedom; just the same, this is a large number of joints if we were to animate every joint by hand.

Each mechanical link, that is, each bone or small set of bones, is assigned a mass and an inertia matrix. These values can be assigned based on those reported in the biomechanics literature; however, in our case we compute them automatically from the geometry of the given model. We set the stiffness of each joint to account for the combined action of muscles and tendons; this models a given level of tension or relaxation about a given pose. This pose is effectively a rest configuration with respect to the equilibrium-point model of human motor control [Fel86, BHMIG92]. Stiffness values estimated by measurement can be found in the literature as well, but we find that a reasonable initial set of values can be obtained by choosing a minimal stiffness that allows the virtual animal to maintain an upright natural rest posture in a simulation with normal gravity. For our quadruped example, this consists of a lower stiffness for the legs, while the back and neck are somewhat stiffer to maintain the head posture. The resulting motions in this case will be in a relaxed style; changing the stiffness and the equilibrium pose will alter the style, but we will discuss this in greater detail later.

Given this physical model consisting of the equilibrium pose, the stiffness of the joints, and the mass of the links, we can use a physically based simulation to evolve its motion over time. We use a standard constrained multibody approach, which we describe in detail in Appendix A; however, let us quickly review that this motion is given by the Newton-Euler equation

$$M\dot{\phi} = w_\phi + w + G^T \lambda. \qquad (1)$$

Here M is the mass matrix, and ϕ and w are block column vectors containing velocities and wrenches (i.e., forces and torques) of all bodies. We express all quantities in the moving body frames, so we must include the wrench w_ϕ due to the current spatial velocities. Joint constraints, and other constraints, require the solution to satisfy $G\phi = 0$, thus Equation 1 also includes constraint forces $G^T \lambda$. Note that w includes wrenches due to gravity, damping, and external forces, but most importantly it includes wrenches due to the displacement of the joints from the equilibrium pose. By defining R to map joint torques to opposing wrenches on body pairs, this wrench has the form $w_k = R^T K(\theta - \theta_0)$, where K is a diagonal matrix containing the stiffness of each joint. Note, R also transforms the body velocities to angular velocities at the joints, $\dot{\theta} = R\phi$.

3.1. Modal Analysis with Constraints

We are not interested in just simulating the dynamics of a virtual animal; we want to generate animations of locomotion. As outlined in the introduction we do this by combining selected low-frequency natural vibrations as basis motions. In this section we briefly describe how we compute the modal analysis of our virtual animals (for more detail we refer the reader to [Sha97]). Note that the analysis we show here differs from that which is typically done in graphics. We deal with constraints and rigid articulated systems in contrast to the analysis of jelly-like objects, trees, and other elastic materials [PW89, Sta97].

Suppose that the compliant joints in our virtual animal's skeleton are in equilibrium. Given a small twist away from equilibrium, ϕh, the wrenches due to the compliant joints will be $R^T K R \phi h$. Ignoring damping for now, we return to Equation 1, but omit the constraint forces, and drop the w_ϕ term since our analysis concerns small low-frequency vibration about the equilibrium with insignificant changes in body frame orientation. Replacing w with this spring force above gives

$$M\dot{\phi} = R^T K R \phi h. \qquad (2)$$

When calculating the vibration modes, we must only take the permissible directions of constrained motion into account. The degrees of freedom of the system are described by the null space of G. For simple chains and trees this null space is straightforward to construct and related to the manipulator Jacobian. However, for loops (e.g., hands holding a basket or pushing a shopping cart) we must solve for the free directions. We use singular value decomposition of G to find

the null space N, which provides the spatial twists that form the minimal set of kinematically admissible directions. Thus, $\phi = N\dot{x}$, where $\dot{x}$ is the velocity in admissible coordinates. Substituting x for $\dot{x}h$, and letting $\dot{\phi} = N\ddot{x}$ (since we are only looking at small vibrations we let $\dot{N} = 0$), we project Equation 2 into these minimal coordinates to obtain

$$\mathsf{M}\ddot{x} = \mathsf{K}x, \qquad (3)$$

with $\mathsf{M} = N^T M N$ and $\mathsf{K} = N^T R^T K R N$. While K will typically be rank deficient, M is always invertible so we can easily solve for the eigenvalues and eigenvectors of Equation 3 as either a generalized eigenvalue problem or via $\mathsf{M}^{-1}\mathsf{K}$. When K is not full rank, the resulting eigenvectors include rigid motions with corresponding eigenvalues equal to zero.

The eigenvectors u_i of $\mathsf{M}^{-1}\mathsf{K}$ are the modes shapes, and can be seen as involving coordinated joint motions RNu_i about the equilibrium pose. The eigenvalues λ_i give the corresponding frequency, $f_i = \sqrt{\lambda_i}/2\pi$ Hz. In this basis, the behaviour of the system separates into a set of independent differential equations. The solution of the entire system is equivalent to the superposition of the solutions to these independent equations. Adding Rayleigh damping to the system (i.e., a damping matrix equal to a combination of the mass and stiffness matrices) does not change the shapes of the modes, but does alter the frequencies of the resulting modes.

Figure 1 depicts the first four modes for a dog model. All but one of the first four modes are parallel to the sagittal plane of the dog and are relevant for locomotion. Mode u_7, however, does not show this symmetry (twist of the whole spine) and as such can be easily rejected as not relevant to locomotion. We will mention more about selecting appropriate modes to generate interesting locomotion animations in Section 4.3.

Finally, note that a joint coordinate formulation of the mass and stiffness matrices would lead us to the same vibration shapes. But our formulation with constraints has the advantage that we can easily add additional constraints in the same framework. For instance, we can easily model a virtual human pushing a cart, or holding an object in two hands; we do this by adding constraints to the hands (forming a closed loop). We can also add biological constraints, such as a constraint to fix the orientation of the head so that it remains stable during locomotion.

4. Creating Locomotion

At this point we know how to compute a set of natural vibration modes for any given animal model. Let us now describe how we can use these modes to create locomotion.

4.1. Dynamic control of simple creatures

Let us start with very simple creatures to see how we can create open loop controllers which produce locomotion for

u_6

u_7

u_8

u_9
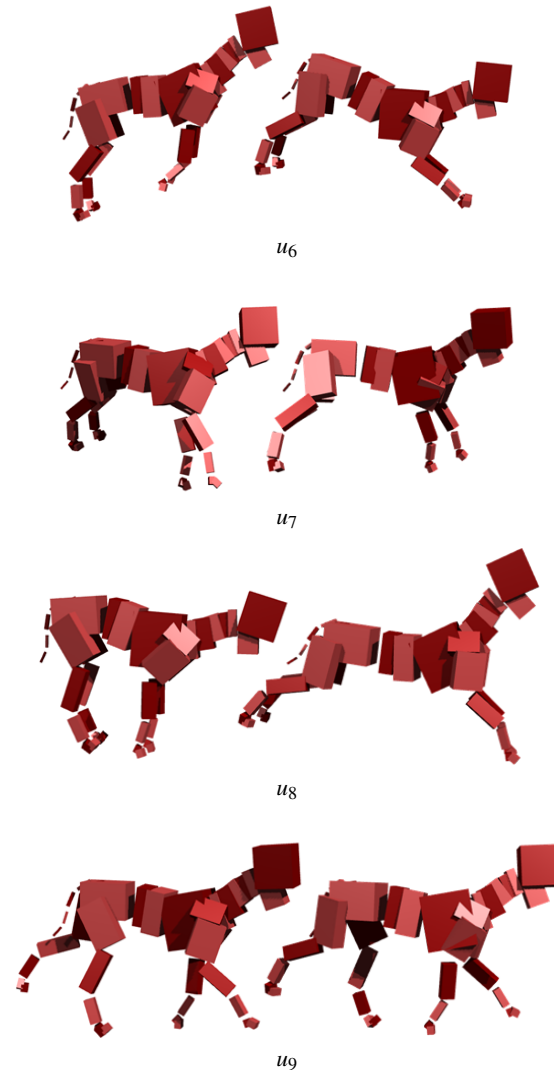
Figure 1: *The first four non rigid vibration modes of a dog model with 80 degrees of freedom. These modes can be described as bounding, back twisting, stretching, and alternating legs, respectively.*

these creatures in a physically based simulation. Consider first the simple example of a loop shaped creature modeled as an articulated chain of identical rigid bodies. Computing vibration modes for such a structure yields vibrations shapes and frequencies depicted in Figure 2 (top). A small push from the muscles in the direction of a particular mode shape will cause the whole body to vibrate with that shape. The resulting vibration is a sinusoid with an exponential decay. Thus, the goal of our controller is to pump energy into some selected modes so that they oscillate without decay at coordinated frequencies and selected phases. In this case,
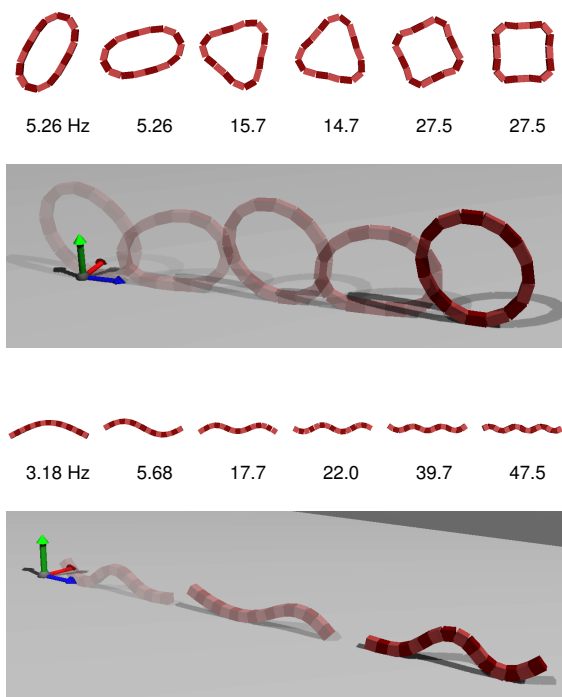
Figure 2: *A simple loop shaped creature (top) and a simple linear creature (bottom) showing each creature's vibration modes along with examples of locomotion obtained with by mixing only the first two modes 90 degrees out of phase of each other.*

the selected modes are the two lowest frequency modes and the controller pushes these at a frequency of 5.26 Hz, but 90 degrees out of phase. Instead of directly applying joint torques, we combine the selected vibration shapes to produce an equilibrium point for use in a PD controller where the stiffness of the controller is exactly that which was used in the analysis. The choice of modes along with their amplitudes and phases in this case are easily adjusted by hand to create an open loop controller that produce locomotion.

Figure 2 (bottom) shows a slightly different simple linear creature. Just like the loop creature, we construct a locomotion controller by combining the two lowest vibration modes, with the second mode shifted by 90 degrees. Note that in this case the vibrations have different frequencies, but this does not pose a serious problem. Damping in the system widens the resonance response of different modes, and likewise contact helps regulate the motion as collisions remove energy from the system.

We have had some success to applying this approach to more complex animals, such as dogs and humans, but ultimately balance becomes a much more significant problem in these cases. Just the same, the natural vibrations of these complex animals still provide an excellent basis for creating kinematic animations of locomotion as we describe in the next section.

4.2. Kinematic animation

The analytic solution of a virtual animal's elastic vibrations provides a straightforward way of creating kinematic animations of locomotion. We can simply move the joints using selected mode shapes modulated by sinusoids, much like the real physical response of the system, except that we can discard the effect of damping. In this case, we also superimpose the selected modes at the frequency of the gait, or in some cases, at twice the frequency in order to animate desirable physical effects. For instance, a head bob might occur on every step, or in the case of a quadruped, its back would normally flex twice during each cycle of a trot.

We can formalize our kinematic control as follows. Note that RNu_i gives the shape of mode i in joint coordinates, and let $\tilde{f}_i$ and γ_i be the selected frequency and phase respectively. Over time, the positions of the joints are set as

$$\theta(t) = \sum_i \alpha_i RNu_i \sin(2\pi\tilde{f}_i t + \gamma_i) + \theta_0. \qquad (4)$$

Here, α_i is a control parameter used to independently tune the amplitude of each mode. Note that only a small set of α_i need to be non-zero in order to create a locomotion, and as such the description is very compact. Note also that Equation 4 is very similar to the equilibrium point used in the previous section, except that here it is used to directly provide the virtual animal's configuration at each point in time.

Our virtual animals have joint limits, and there are naturally situations where these limits are reached. For example, if the rest pose has joints near their limits then oscillation around this pose may exceed the limits. We address this by simply modifying the angles produced by Equation 4 when they exceed their limits. To avoid joint popping artefacts that result from simply clamping joint angles to their limits, we instead impose the limits in a soft manner. We let each joint exceed its limits at most by a small amount $\Delta\theta_{soft}$. Any motion of a joint past its limit, which we denote d, is reduced by $d^2/(d + \Delta\theta_{soft})$.

While our kinematic controllers constrain joint motion to respect joint limits, they do not currently treat ground contact. Layering an inverse kinematics solution on top of this motion is one possible solution that we leave for future work.

4.3. Parameter selection

Let us now describe how we choose mode combinations to animate locomotion of complex virtual animals. Recall that we focus on the low-frequency modes since they require the least energy to produce and maintain large deformations of the structure. The frequencies of these modes are also appropriate for locomotion. For all the virtual animals we used

in this work, we observe that the lower modal frequencies always correspond well with the period of real gaits.

The heuristics for selecting important modes are simple. We chose modes that involve leg swings in the direction of movement, and knee or elbow bending. Modes with jumping jack motions and body twists are easily discarded (though perhaps useful for other kinds of animation). Phases can also be automatically assigned the integer multiples of 90 degrees that produces a forward cyclic motion useful for locomotion. Specifically, for walking and running we select modes that exhibit maximal alternating knee bends (A), alternating forward motion of the feet (B), and alternating vertical motion of the feet (C). Knee bending and vertical motion modes are given a phase offset of 90 degrees and we automatically enable the top ranked mode in each category. For jumping, we similarly identify modes that exhibit maximal synchronous knee bends (D), and synchronous vertical motion of the feet (E). Likewise, for quadrupeds we also include heuristics that identify bending (F) and swinging (G) of the front legs. Finally, the magnitudes can be easily tuned to adjust the motion (e.g., increase stride length). Figure 3 summarizes a number of different animations that can be easily produced with these heuristics.

Note that the equilibrium pose around which the modal vibrations are computed has an influence on the resulting locomotion. Figure 3 shows an example of this where a T-pose can be used to create a walk with the arms extended, while a rest pose involving bent knees and bent elbows is more suitable for a run.

While we can alter the rest pose as a means of modifying the style of an animation, we can also alter the style of a locomotion animation by using the mode shapes for postural changes. That is, we can add a term $\beta_i RNu_i$ to Equation 4 and set some small number of β_i to non-zero. An initial walking motion can be easily modified so that the legs are closer together, or so that arms are raised or lowered. Examples are shown in the accompanying video.

Lastly, layering mode shapes to build locomotion constrains the user to select motions from an intuitive palette of natural vibrations, ensuring plausibility and leading to low energy motions. Indeed there are still other important factors beyond mechanical efficiency that effect how an animal moves, including, for example, constraints on posture to improve sensory input from the eyes or the ears. Figure 3, shows an example where head rotation was controlled with a constraint, and thus the modal basis consists of motions that also respect this constraint.

5. Results and Discussion

The accompanying video shows examples of dynamic locomotion for the simple loop shaped and linear creatures, as well kinematic animations of dog and human models for the

actions described in Figure 3. The video also shows examples of how it is easy to alter the posture of these motions by adding an offset of a low-frequency mode shape. For instance, the jumping jack animation can be modified so that the character appears to be supporting more of the weight on one leg. The last examples show the benefit of using additional constraints to animate a running human with a stable head, and a walking human pushing an imaginary cart.

Combining a small number of basis motions is an attractive method for creating animation. The natural vibrations give us an intuitive palette for creating animations of cyclic motions such as locomotion. In comparison, building motion curves for individual joints is very time consuming. Instead, we can quickly adjust a small set of automatically identified basis motions that capture important and natural physically based movements of the virtual character. This is the central contribution of our approach. This work also leads in a number of interesting directions for future work, for example, the development of motor control of fully dynamic virtual animals that builds on modal vibrations but also correctly treat ground contact and balance.

Limitations: It is important to correctly set stiffness for proper control in a dynamic simulation, but this is less important in producing a useful palette of global body motions. Changing the stiffness will alter the frequency, and some modes may have their frequencies aligned as a result. For example, in the quadruped model with uniform stiffness, lowering the stiffness on the back legs results in modal vibrations where all legs (front and back) move at once in two different modes. Regardless of how we might tune the stiffnesses, all the necessary motions for building a locomotion animation are available and can automatically be identified with heuristics. While altering the stiffness does change the exact mode shapes, this can be seen as adjusting the style of the basis motions. For example, a stiff leg will alter the leg swinging modes, but the increased stiffness of this leg will also be present in all the other modal vibrations, as well as final motions that we create.

6. Conclusion

This paper has revisited motion control of virtual animals by expressing the degrees of freedom for a controller in a new basis, the space of low-frequency vibration modes of the underlying mechanical model. This brings a number of benefits. Firstly, if associated with a good balance controller, vibration modes could be used to generate controllers for physically based motions of complex animals. Just the same, they provide a good basis for creating kinematic animations of locomotion since they involve simultaneous motion at all joints, and have a natural appearance and a physical justification. For this same reason, the modes serve as intuitive building blocks for tuning these animations. Because we limit our search to low-frequency modes and identify important modes automatically with heuristics, the search space for

model / action	DOFs	rest pose	stiffness	constraints	parameters				sample frame
loop creature	19		1 N/rad uniform	none	$\alpha_6= 1$ $\gamma_6= 0$ $\tilde{f}_6= 5.25$ $\alpha_7= 1$ $\gamma_7= 1.57$ $\tilde{f}_7= 5.25$				
linear creature	21		1 N/rad uniform	none	$\alpha_6= 1$ $\gamma_6= 0$ $\tilde{f}_6= 5$ $\alpha_7= 1$ $\gamma_7= 1.57$ $\tilde{f}_7= 5$				
dog trot	80		20 N/rad spine ×5	none	$\alpha_9 = 0.5$ $\gamma_9 = 0$ $\tilde{f}_9 = 2$ $\alpha_{11}= 0.5$ $\gamma_{11}= 0$ $\tilde{f}_{11}= 2$ $\alpha_{19}= 0.5$ $\gamma_{19}= 1.57$ $\tilde{f}_{19}= 2$ $\alpha_{21}= 0.5$ $\gamma_{21}= 1.57$ $\tilde{f}_{21}= 2$				
dog amble	80		20 N/rad spine ×5	none	$\alpha_{10}= 0.5$ $\gamma_{10}= 0$ $\tilde{f}_{10}= 2$ $\alpha_{11}= 0.1$ $\gamma_{11}= 0$ $\tilde{f}_{11}= 2$ $\alpha_{19}= 0.5$ $\gamma_{19}= 1.57$ $\tilde{f}_{19}= 2$ $\alpha_{21}= 0.3$ $\gamma_{21}= 1.57$ $\tilde{f}_{21}= 2$				
dog gallop	80		20 N/rad spine ×5	none	$\alpha_4 = 0.2$ $\gamma_4 = 0$ $\tilde{f}_4 = 2$ $\alpha_6 = 1.0$ $\gamma_6 = 1.57$ $\tilde{f}_6 = 2$ $\alpha_8 = 1.0$ $\gamma_8 = 0$ $\tilde{f}_8 = 2$ $\alpha_{10}= 0.2$ $\gamma_{10}= 1.57$ $\tilde{f}_{10}= 2$ $\alpha_{11}= 0.7$ $\gamma_{11}= 1.57$ $\tilde{f}_{11}= 2$ $\alpha_{22}= 1.0$ $\gamma_{22}= 1.57$ $\tilde{f}_{22}= 2$				
human T-pose walk	47		100 N/rad spine ×10	none	$\alpha_6 = 0.5$ $\gamma_6 = 0$ $\tilde{f}_6 = 2$ $\alpha_{17}= 0.76$ $\gamma_{17}= 1.57$ $\tilde{f}_{17}= 2$				
human jumping	47		100 N/rad spine ×10	none	$\alpha_4 = 0.75$ $\gamma_6 = 0$ $\tilde{f}_6 = 2$ $\alpha_{16}= 0.75$ $\gamma_{16}= 0$ $\tilde{f}_{16}= 2$ $\alpha_{24}= 0.75$ $\gamma_{24}= 0$ $\tilde{f}_{24}= 2$				
human jumping jacks	47		100 N/rad spine ×10	none	$\alpha_4 = 0.75$ $\gamma_6 = 0$ $\tilde{f}_6 = 2$ $\alpha_7 = 0.35$ $\gamma_7 = 0.79$ $\tilde{f}_7 = 1$ $\beta_7= -0.3$ $\alpha_{10}= 0.95$ $\gamma_{10}= 0.79$ $\tilde{f}_{10}= 1$ $\alpha_{16}= 0.75$ $\gamma_{16}= 0$ $\tilde{f}_{16}= 2$ $\alpha_{24}= 0.75$ $\gamma_{24}= 0$ $\tilde{f}_{24}= 2$				
human run	47		100 N/rad spine ×10	none	$\alpha_6 = 0.5$ $\gamma_6 = 0$ $\tilde{f}_6 = 2$ $\alpha_{13}= 0.5$ $\gamma_{13}= 1.57$ $\tilde{f}_{13}= 2$ $\alpha_{15}= 0.75$ $\gamma_{15}= 1.57$ $\tilde{f}_{15}= 2$				
human run	44		100 N/rad uniform	head	$\alpha_5 = 0.5$ $\gamma_5 = 0$ $\tilde{f}_5 = 2$ $\alpha_7 = 0.5$ $\gamma_7 = 0$ $\tilde{f}_7 = 2$ $\alpha_{13}= 0.5$ $\gamma_{13}= 1.57$ $\tilde{f}_{13}= 2$ $\alpha_{14}= 0.5$ $\gamma_{14}= 1.57$ $\tilde{f}_{14}= 2$				
human pushing cart	42		100 N/rad spine ×10	hands	$\alpha_2 = 0.5$ $\gamma_2 = 0$ $\tilde{f}_2 = 2$ $\alpha_9 = 0.5$ $\gamma_9 = 1.57$ $\tilde{f}_9 = 2$ $\alpha_{10}= 0.5$ $\gamma_{10}= 1.57$ $\tilde{f}_{10}= 2$				

Figure 3: *An overview of different animation parameters for a variety of animals, gaits and situations. Heuristics (A) through (F) were used to identify the modes in all cases except the linear and loop shaped and creatures, the dog gallop, and the human jumping jacks.*

the animation parameters is greatly simplified, especially in the case of complex virtual animals. Finally, our method for finding these mode shapes is specially tuned to the case of skeletal animals; the bones are rigid while the joints are elastic, and we allow for the creation of additional constraints as desired.

Appendix A: Constrained Rigid Body Dynamics

We use a standard method for modelling and simulating a constrained system of rigid bodies (for an explanation in depth, see [Fea87] or [MLS94]).We place the local frame for body *a* at the body's center of mass and with axes aligned with the principle axes of inertia. The homogeneous coordi-

nates of frame *a* with respect to frame *b* is given by the 4×4 matrix

$$_a^b\mathsf{E} = \begin{pmatrix} \Theta & p \\ 0 & 1 \end{pmatrix},$$

where Θ is a 3×3 rotation matrix, and p is a 3×1 displacement. We write the spatial velocity of body *a* with respect to the world frame *W* as a column vector ${}^a\phi_{a-W} = (\omega^T, v^T)^T$, where ω is the angular velocity and v is the linear velocity. The leading superscript denotes that the vector is expressed in the coordinates of body *a*. Spatial forces, called wrenches, are written ${}^aw = (\tau^T, f^T)^T$ where τ is the rotational torque and f is the translational force. Spatial velocities transform according to the adjoint transformation ${}_a^b\mathsf{A}d$, wrenches transform with the inverse transpose, i.e., ${}^bw = {}_b^a\mathsf{A}d^T\, {}^aw$. The 6×6 adjoint matrix is defined by

$$_a^b\mathsf{A}d = \begin{pmatrix} \Theta & 0 \\ [p]\Theta & \Theta \end{pmatrix},$$

where $[p]$ denotes the skew symmetric 3×3 matrix equivalent to the cross product $p\times$.

The motion of a set of rigid bodies is given by the Newton-Euler equation

$$M\dot{\phi} = w_\phi + w. \tag{5}$$

Here *M* is a diagonal mass matrix assembled for all bodies, and we let ϕ and *w* (without leading superscripts) refer to the block colum vectors containing velocities and wrenches of all bodies. Since we express these quantities in the moving body frames, we must include the wrench w_ϕ due to the current spatial velocities. Lastly, *w* represents wrenches due to gravity, muscle stiffness, and damping.

Following [BW98], we discretize the system by replacing the acceleration $\dot{\phi}$ with the approximation $(\phi^{t+h} - \phi^t)/h$ and solve directly for the spatial velocities after a small time step *h*. Stiff forces in *w* are made implicit by replacing *w* with a Taylor approximation of its value at the next time step and moving $T\phi^{t+h}$, the terms involving ϕ^{t+h}, to the left hand side.

$$(M - hT)\phi^{t+h} = h(w_\phi + w) + M\phi^t. \tag{6}$$

Constraints

The bones in the skeleton of our animal models must stay connected and can only rotate about certain axes. These constraints change Equation 5 into a differential algebraic equation with a constraint for each joint, $g_j = 0$ (a function of the current state). We use a standard technique for solving the constrained equation [AP98]; we differentiate the constraint once to turn it into a constraint on the velocity, and then we stabilize the solutions to prevent numerical drift.

We define a frame *j* for each joint, located at the joint, and with the *x* axis purposely aligned with the free rotation for rotary joints. The velocity constraint for joint *j* connecting body *a* and body *b* has a very simple form when the relative velocity of *a* and *b* is written in the coordinates of the frame *j*. For a single axis rotary joint, it states that the velocity must be zero in all directions exept for rotation about the *x* axis, i.e.,

$$I_{2:6}\, {}^j\phi_{a-b} = 0, \tag{7}$$

where $I_{2:6}$ is the 5×6 matrix consisting of the bottom 5 rows of the size 6 identity matrix and ${}^j\phi_{a-b} = {}_a^j\mathsf{A}d\, {}^a\phi_{a-W} - {}_b^j\mathsf{A}d\, {}^b\phi_{b-W}$. For a spherical joint we use $I_{4:6}$. Written in matrix form the constraints require $G\phi$ to be zero, where the sparse matrix *G* has 3 rows for every ball joint, and 5 rows for every hinge joint. To solve for the new constrained spatial velocities, we solve the Lagrangian,

$$\begin{pmatrix} (M - hT) & G^T \\ G & 0 \end{pmatrix} \begin{pmatrix} \phi^{t+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} h(w + w_\phi) + M\phi^t \\ b \end{pmatrix}, \tag{8}$$

where *b* provides Baumgarte stabilization.

Ground contacts and joint limits are unilateral constraints. While these constraints could be properly accounted for in simulations by setting up a linear complementarity problem, we find that intermittent stiff implicit penalty forces are sufficient for our virtual animals. For modal analysis, however, we can compute modes involving ground contacts by including a bilateral constraint for the contacts.

Compliant Joints

The muscles and tendons of an animal produce torques at its joints. We model this with spring and damping torques. It is the inclusion of stiffness in our constrained rigid bodies that allows elastic oscillations in the system, for which we use modal analysis to determine which vibrations are the most important.

The torque at a compliant single axis joint with stiffness k_j is ${}^j\tau_{jx} = k_j(\theta_j - \theta_{j0})$, where θ_j is the current joint angle and θ_{j0} is the desired angle. The torque acting upon each of the connected bodies is equal but opposite: ${}^aw_k = {}_a^j\mathsf{A}d^T I_{1:1}^T\, {}^j\tau_{jx}$ and ${}^bw_k = -{}_b^j\mathsf{A}d^T I_{1:1}^T\, {}^j\tau_{jx}$. Here the 1×6 matrix $I_{1:1}$ is the first row of the identity matrix and corresponds with the degrees of freedom that were not constrained in Equation 7. In a similar manner to *G*, we construct a matrix *R* which allows the spring wrenches to be written in matrix form, $w_k = R^T K(\theta - \theta_0)$. Here *K* is the diagonal stiffness matrix containing the stiffness of each joint. Note that *R* also transforms the spatial velocities of the bodies to angular velocities at the joints, $\dot{\theta} = R\phi$. Thus, damping wrenches can be computed as $w_c = R^T C R\phi$, where *C* is the diagonal matrix of damping coefficients.

References

[Ale88] ALEXANDER R. M.: *Elastic Mechanisms in Animal Movement*. Cambridge University Press, 1988.

[Ale96] ALEXANDER R. M.: *Optima for Animals*. Princeton University Press, 1996.

[AP98] ASCHER U. M., PETZOLD L. R.: *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.

[BHMIG92] BIZZI E., HOGAN N., MUSSA-IVALDI F. A., GISZTER A.: Does the nervous system use equilibrium-point control to guide single and multiple joint movements? *Behavioral and Brain Sciences 15* (1992), 603–613.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), pp. 43–54.

[CRTW05] COLLINS S., RUINA A., TEDRAKE R., WISSE M.: Efficient bipedal robots based on passive-dynamic walkers. *Science 307* (2005), 1082–1085.

[CWR01] COLLINS S. H., WISSE M., RUINA A.: A three-dimensional passive-dynamic walking robot with two legs and lees. *The International Journal of Robotics Research 20*, 2 (2001), 607–615.

[Fea87] FEATHERSTONE R.: *Robot dynamics algorithms*. Kluwer, 1987.

[Fel86] FELDMAN A. G.: Once more on the equilibrium-point hypothesis (lambda model) for motor control. *Journal of Motor Behavior 18*, 1 (March 1986), 17–54.

[FRDC04] FAVREAU L., REVERET L., DEPRAZ C., CANI M.-P.: Animal gaits from video. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), pp. 277–286.

[FvdPT97] FALOUTSOS P., VAN DE PANNE M., TERZOPOULOS D.: Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics 3*, 3 (September 1997), 201–214.

[GT95] GRZESZCZUK R., TERZOPOULOS D.: Automated learning of muscle-actuated locomotion through control abstraction. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), pp. 63–70.

[GTH98] GRZESZCZUK R., TERZOPOULOS D., HINTON G.: Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of SIGGRAPH 98* (1998), pp. 9–20.

[HP97] HODGINS J. K., POLLARD N. S.: Adapting simulated behaviors for new characters. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 153–162.

[HRE*08] HECKER C., RAABE B., ENSLOW R. W., DEWEESE J., MAYNARD J., VAN PROOIJEN K.: Real-time motion retargeting to highly varied user-created morphologies. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–11.

[HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM Press, pp. 71–78.

[Ijs08] IJSPEERT A. J.: Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks 21*, 4 (2008), 642–653.

[Kuo02] KUO A. D.: Energetics of actively powered locomotion using the simplest walking model. *Journal of Biomechanical Engineering*, 124 (2002), 113–120.

[LHP05] LIU C. K., HERTZMANN A., POPOVIC Z.: Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics 24*, 3 (2005), 1071–1081.

[LP02] LIU C. K., POPOVIC Z.: Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 408–416.

[MA90] MCGEER T., ALEXANDER R.: Passive bipedal running. *Proceedings of the Royal Society of London, Series B, Biological Sciences 240* (1990).

[McG90] MCGEER T.: Passive dynamic walking. *International Journal of Robotics Research 9*, 2 (1990), 62–82.

[MLS94] MURRAY R., LI Z., SASTRY S. S.: *A mathematical introduction to robotic manipulation*. CRC Press, 1994.

[MZ90] MCKENNA M., ZELTZER D.: Dynamic simulation of autonomous legged locomotion. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (1990), pp. 29–38.

[NF02] NEFF M., FIUME E.: Modeling tension and relaxation for computer animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2002), pp. 81–88.

[PW89] PENTLAND A., WILLIAMS J.: Good vibrations: modal dynamics for graphics and animation. In *SIGGRAPH '89* (1989), ACM Press, pp. 215–222.

[RH91] RAIBERT M. H., HODGINS J. K.: Animation of dynamic legged locomotion. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (1991), pp. 349–358.

[Sha97] SHABANA A. A.: *Vibration of Discrete and Continuous Systems*. Springer, 1997.

[SHP04] SAFONOVA A., HODGINS J. K., POLLARD N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM Transactions on Graphics 23(3), SIGGRAPH 2004* (2004).

[Sim94] SIMS K.: Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM Press, pp. 15–22.

[SM01] SUN H. C., METAXAS D. N.: Automating gait generation. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 261–270.

[Sta97] STAM J.: Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum 16*, 3 (1997).

[Tag95] TAGA G.: A model of the neuro-musculo-skeletal system for human locomotion. *Biological Cybernetics*, 73 (1995), 97–1995.

[TvdP98] TORKOS N., VAN DE PANNE M.: Footprint–based quadruped motion synthesis. In *Graphics Interface* (1998), pp. 151–160.

[vdP96] VAN DE PANNE M.: Parameterized gait synthesis. *IEEE Computer Graphics and Applications. 16*, 2 (1996), 40–49.

[vF93] VAN DE PANNE M., FIUME E.: Sensor-actutor networks. In *Computer Graphics Proceedings, ACM SIGGRAPH, Proceedings of SIGGRAPH '93* (1993), pp. 335–342.

[vKF94] VAN DE PANNE M., KIM R., FIUME E.: Virtual wind-up toys for animation. In *Proceedings of Graphics Interface '94* (Banff, Alberta, Canada, 1994), pp. 208–215.

[Wil97] WILHELMS J.: Animals with anatomy. *IEEE Comput. Graph. Appl. 17*, 3 (1997), 22–30.

[WP03] WU J.-C., POPOVIC Z.: Realistic modeling of bird flight animations. *ACM Trans. Graph. 22*, 3 (2003), 888–895.

$$\therefore$$

5.4 WIND PROJECTION BASIS FOR REAL-TIME ANIMATION OF TREE

J. Diener, M. Rodriguez, L. Baboud, L. Reveret Computer Graphics Forum,
Proceedings of Eurographics'09, Münich, Germany, april 2009.

Wind projection basis
for real-time animation of trees

Julien Diener[1], Mathieu Rodriguez[2,3], Lionel Baboud[1], Lionel Reveret[1]

[1] Laboratoire Jean Kuntzmann, INRIA Rhône-Alpes
[2] Département de mécanique, LadHyX, Ecole Polytechnique-CNRS
[3] UMR547 PIAF, INRA, Université Blaise Pascal

Figure 1: *A single tree and a forest animated with our method.*

Abstract

This paper presents a real-time method to animate complex scenes of thousands of trees under a user-controllable wind load. Firstly, modal analysis is applied to extract the main modes of deformation from the mechanical model of a 3D tree. The novelty of our contribution is to precompute a new basis of the modal stress of the tree under wind load. At runtime, this basis allows to replace the modal projection of the external forces by a direct mapping for any directional wind. We show that this approach can be efficiently implemented on graphics hardware. This modal animation can be simulated at low computation cost even for large scenes containing thousands of trees.

1. Introduction

Vegetation is an important part of the realistic depiction of natural scenes in a virtual environment. While research in plant modeling has produced impressive results, animation techniques remain of visually poor quality, especially for real-time applications.

The current development of computer hardware allows real-time applications to render natural scenes with high quality geometric details. However, the increase in model complexity also generates the need for suitable animation techniques. As the existing accurate simulation methods are still too computationally expensive, developers usually resort to ad-hoc techniques. Such methods can be satisfying

for simple models such as grass or palm trees, but fails to provide realistic animation of complex branching structures such as standard trees.

Similarly to [Sta97], we rely on a modal analysis to animate 3D models of trees. This technique simulates the deformation of the plant using a precomputed deformation basis. While it provides visually convincing results, it cannot be used in an interactive application as it requires full integration of the dynamic wind forces over the entire plant at runtime. Real-time framerates can be obtained only for very simple models but not for realistically complex trees.

Our contribution is a method that makes the computation of the wind forces integration almost costless and indepen-

dent of the tree complexity. The only remaining computations at runtime consist in the update of the positions of animation control points, defining the skeleton used to animate the rendered geometry. To this end we define a modal projection basis where the influence of directional wind loads can be precomputed off-line.

After introducing modal animation of trees in section 3, we show in section 4 how the wind projection basis is computed and how it is used to avoid the integration of wind forces at runtime. Finally some implementation issues are discussed in section 5. We show that an entire forest containing thousands of trees can be animated in real time using graphics hardware, producing realistic results for both close and distant trees.

2. State of the art

Many models defined for real-time animation of trees use mechanical simplification specifically designed for that purpose. For example, in [OTF*04] and [AK06] the tree dynamics is modeled using the *static response* of each branch segment (*i.e.* which does not consider time integration of the dynamics). However both methods focus instead on the wind formulation. Ota et al. propose a realistic stochastic description. Akagi et al. incorporate the influence of the plant in the computation of the wind flow modeled with the Navier-Stokes equation.

To animate forests in real-time, Di Giacomo et al. [GCF01] also use static response to animate the tree's terminal branches. For the rest of the structure they simulate the dynamics of the rotation angles between branches using an explicit Euler method, in a similar way as Sakaguchi and Ohya in [SO99].

These previous works model the tree's mechanical response expressed in generalized coordinates. However they solve the dynamics locally (i.e segment by segment) and do not fully model the inertia of the subsequent branches that are displaced by a joint's rotation. Simulation techniques of global mechanical structures such as [Bar96] provide correct dynamics without this limitation. Weber [Web08] developed such a simulation method adapted to trees.

For real-time application only little computation time is committed to the animation of vegetation. The use of parallelizable algorithms has become essential. However animation described by generalized coordinates makes the computation of nodes position hierarchically dependent on each other. Weber proposes a scheme to use multithreading, but it is not yet appropriate for highly parallel computing provided by graphics processing unit (GPU).

Data driven methods are interesting alternatives to direct simulation. Diener et al. [DRF06] use video to extract motion data from real plants and reproduce it on virtual models. James et al. in [JF03] and [JTCW07] use precomputed simulation as input to generate realistic response to external impulse and to fluctuating wind respectively. However the main disadvantage of these approaches is the restricted controls at run time.

First introduced to the computer graphic community by Pentland and Williams [PW89], modal analysis has proven to be a powerful basis for the simulation of oscillatory behaviors. It has then be used for many applications such as precomputed response in an animation using graphics hardware [JP02], real-time collision and manipulation [HSO03] and even sound generation [JBP06].

Modal analysis provides a basis of deformations where all elements of the basis are independent harmonic oscillators. For trees, Shinya and Fournier [SF92] applied this approach to model each of the branches. Stam [Sta97] was the first to use modal decomposition of the entire structure to animate plants. The main idea is to generate a stochastic wind and compute the modal dynamics directly in the frequency domain using the specific response of harmonic oscillators. The goal of this method is to precompute a plant's motion. To use this technique with an interactive wind its modal projection needs to be computed at runtime (see section 4). It would be conceivable on today's graphic hardware but only for coarse tree structures.

Our contribution consists in a method that drastically reduces the computations needed to simulate modal dynamics under controlable wind load. This is done by precomputing a basis of the wind modal projections. Moreover the dynamics is simulated with a computational cost independent of the tree structure complexity and directly proportional to the required quality of the animation. This makes the implementation of level of detail straightforward: the number of animation modes and the subset of tree nodes necessary for rendering can be adjusted freely at runtime.

3. Modal animation framework

In this section, we briefly describe the mechanical model. We introduce modal analysis and explain how it can be used in a real-time animation framework.

Mechanical modeling and modal analysis have been thoroughly studied. We advise interested readers to refer to specialized literature such as [GR94].

3.1. Mechanical model

To solve dynamics over time we resort to numerical simulation applied on a discretized model of a tree obtained using the Finite Element Method (FEM). The principle is to decompose the mechanical system into a set of *elements* separated by *nodes*. The continuous system's deformation is defined as a specific interpolation between nodes displacements.

In our case, as in [Sta97], the system domain Ω is the *lineic* representation of the tree, i.e. the set of central axes

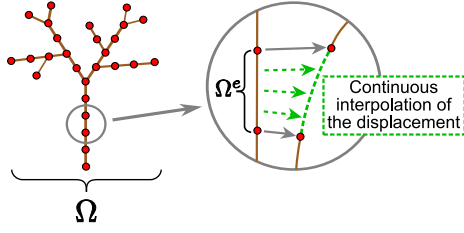of the branches. Then the FEM discretizes Ω as a set of segments Ω^e (see figure 2).

Figure 2: *Finite element discretization.*

Let $\mathbf{u}(t)$ be the vector containing the displacements of all the nodes at time t. The dynamics of a dissipative system is commonly expressed by the equation of motion:

$$M\ddot{\mathbf{u}}(t) + C\dot{\mathbf{u}}(t) + K\mathbf{u}(t) = \mathbf{f}(t) \tag{1}$$

where $\mathbf{f}(t)$ is the external force (see section 4), M and K are the mass and stiffness matrices. The commonly accepted model for the damping matrix C is a linear combination of M and K :

$$C = \alpha M + \beta K \tag{2}$$

More details on the finite element representation and the matrices used in equation (1) are given in appendix.

3.2. Modal analysis

Modal analysis is the decomposition of the deformations of a mechanical system into a set of special deformations called *vibration modes*. These modes are the solutions $\mathbf{u}(t)$ of the free vibration equations:

$$M\ddot{\mathbf{u}}(t) + K\mathbf{u}(t) = 0 \tag{3}$$

$$\mathbf{u}(t) = \lambda_i(t) \cdot \varphi_i \tag{4}$$

where the φ_i are constant vectors called the *modal deformations*, and the $\lambda_i(t)$ are scalar functions of time. Substituting (4) in (3) gives:

$$M^{-1}K\varphi_{\mathbf{i}} = \mu_i\varphi_{\mathbf{i}} \quad \text{with} \quad \mu_i = -\frac{\ddot{\lambda}_i(t)}{\lambda_i(t)} \tag{5}$$

As K, M and φ_i are constants, μ_i are constant scalars.

Finally, the eigenvectors and eigenvalues of $M^{-1}K$ are the solutions of equation (5). Let Φ be the matrix containing all the modal deformations φ_i, and $\mathbf{q}(t)$ the expression of vector $\mathbf{u}(t)$ in eigenspace:

$$\Phi\mathbf{q}(t) = \mathbf{u}(t) \tag{6}$$

Substituting (6) in equation (1) and multiplying on the left by Φ^T, we get:

$$\underline{M}\ddot{\mathbf{q}}(t) + \underline{C}\dot{\mathbf{q}}(t) + \underline{K}\mathbf{q}(t) = \Phi^T \mathbf{f}(t) \tag{7}$$

where $\underline{M} = \Phi^T M\Phi$, $\underline{C} = \Phi^T C\Phi$ and $\underline{K} = \Phi^T K\Phi$. A classical

result of modal analysis is that these matrices are diagonal [GR94]. We can then rewrite equation (7) as $6n$ independent equations such that, for each element q_i of vector $\mathbf{q}$:

$$\ddot{q}_i(t) + \gamma_i\dot{q}_i(t) + \omega_i^2 q_i(t) = \frac{1}{m_i}f_i(t) \tag{8}$$

$$\text{with} \quad \begin{cases} \omega_i^2 = \dfrac{k_i}{m_i} \\ \gamma_i = \alpha + \beta * \omega_i^2 \end{cases} \tag{9}$$

where $f_i(t)$ is the *modal force* induced by the wind (see section 4), ω_i the natural frequency, γ_i the damping coefficient of mode i, m_i and k_i the modal mass and damping respectively (i.e the diagonal elements of $\underline{M}$ and $\underline{K}$).

3.3. Animation using the modes of deformation

Modal analysis is well suited to our purpose because it extracts the most representative components of the structure deformations. For the case of plants, the vibration modes associated with low frequencies (ω_i) have the most significant (and largely distributed) influence on the final motion [RdLM08]. Typically the first modes (of lower frequencies) are large deformations of the whole tree. Then as the modal frequencies increase, the deformation becomes more and more confined on the last tiny branches (see figure 4). Hence only a small subset of the mode matrix Φ needs to be kept in order to model most of the motion complexity.

Equation (8) means that each mode behaves as a one-dimensional harmonic oscillator (see figure 3). Each vibration mode can be seen as a deformation of the whole structure oscillating independently from the others.
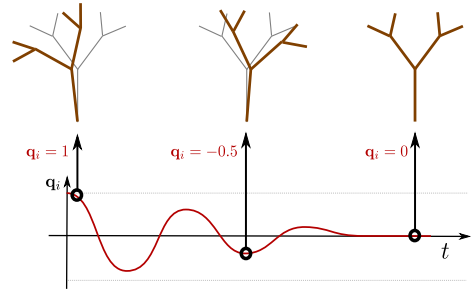
Figure 3: *A vibration mode is a deformation of the whole tree behaving as a harmonic oscillator.*

At each time step, the simulation of modal animation consists in the following procedure:

1. Compute $f_i(t)$, the modal projection of the wind.
2. Update the dynamics of the modes using the explicit Euler method:

$$\begin{aligned} \ddot{q}_i^{t+\delta t} &\leftarrow f_i(t)/m_i - \gamma_i\dot{q}_i^t - \omega_i^2 q_i^t \\ \dot{q}_i^{t+\delta t} &\leftarrow \dot{q}_i^t + \delta t\,\ddot{q}_i^{t+\delta t} \\ q_i^{t+\delta t} &\leftarrow q_i^t + \delta t\,\dot{q}_i^{t+\delta t} \end{aligned}$$

3. Compute the updated displacement:

$$\mathbf{u}^{t+\delta t} = \sum_{\forall i} q_i^{t+\delta t} \varphi_i \qquad (10)$$

Finally, only the evaluation of the modal forces $f_i(t)$ requires costly computations. This is the main difficulty that our technique proposes to resolve. To this end we now describe a basis that can be computed off-line and avoids costly computations at runtime.

4. Wind projection basis

Integrating a general wind load over the whole tree requires a sum over all the elements (branches) at each time step of the animation (as done by [Sta97]). Doing so prevents real-time animation of sufficiently complex trees. By making the assumption that the wind load is uniform over the whole tree, a projection basis can be precomputed that allows to animate the tree in real-time under the load of any directional wind.

Note that the wind is considered uniform only over each tree but can vary between distinct instances. The speed and direction of the wind are left unconstrained and can be freely controlled at runtime.

We start by modeling the drag load induced by the contact of wind on a rigid structure. As in [SFL06, dL08], we model the wind load on any point $p \in \Omega$ at time t by:

$$\mathbf{f}(p,t) = \frac{1}{2} \rho C_D D(p) \|\mathbf{v}(p,t) - \dot{p}\| (\mathbf{v}(p,t) - \dot{p}) \qquad (11)$$

where $\mathbf{v}(p,t)$ is the wind speed vector and $D(p)$ is the diameter of the branch at point p. The air density ρ and the drag coefficient C_D, depending on the shape of the branches, are constant. We merge them as one scalar factor C.

From this local definition, one can derive the modal wind load for each mode i by modal projection [GR94]:

$$f_i = \int_\Omega \mathbf{f}(p) \cdot \varphi_i(p) dp \qquad (12)$$

where the time parameter is omitted for clarity and $\varphi_i(p)$ is the vector containing the ith modal deformation at p.

We do not take into account the influence of leaves and the direction of branches in equation (11). The drag of a single leaf is a complex phenomenon still not well understood. We observed that taking it into account using an isotropic drag model (that can be seen as a statistical mean) does not does not have a significant influence on the modal decomposition. Thus we decided to omit it in our computations. For branches, wind forces should be projected taking into account branches directions. Because the modes with low frequency are mostly perpendicular to the branches, the modal projection (the dot product of $\mathbf{f}$ and φ in equation (12)) has a similar effect.

In the case of a freely defined finite element discretization, equation (12) is computed as the sum of the elements

contribution:

$$f_i = \sum_e \int_{\Omega_e} \mathbf{f}(p) \cdot R^e N^e(p) \varphi_i^e dp \qquad (13)$$

where R^e is the rotation matrix mapping the local frame of the element e to the global frame. φ_i^e is the vector containing the ith modal deformation of both element's border nodes (in the local frame) and $N^e(p)$ is the *shape function matrix* defining the interpolation between these nodes (see appendix).

To compute the modal forces, equation (12) can be discretized over the nodes of the FEM (in the frequency domain) as in [Sta97]:

$$\widehat{f_i} = \sum_{n \in \Omega} \widehat{f}(n) \cdot \varphi_i(n)$$

For simple structure it is a suitable approach but for complex trees such a modal projection it would be prohibitive. Using a simplified structure is not a solution as it would result in a poor approximation of the effective modal stress.

In our method, to reduce the cost of the modal projection of the wind forces at runtime, we focus on extracting the time dependent parameters out of the integral. The remaining part is then precomputed.

The first simplification is to consider R^e as a constant: the wind load is computed on the tree in its rest position. The orientation changes of the branches is not significant as they remain small for natural trees motion. Compensatory models were tested but did not bring noticeable improvement. With this approximation, the modal deformation term is constant and only the wind force depends on time. To extract runtime parameters, we assume a wind model such that:

1. From the wind force, we only keep the *mean aerodynamic stress*:

$$\mathbf{f}(p) = C D(p) \|\mathbf{v}(p,t)\| \mathbf{v}(p,t)$$

The remaining part of the external force, called the *aerodynamic damping*, can be compensated by an increase of the damping coefficients α and β from equation (2).

2. The wind has a spatially uniform speed $\mathbf{v}(t)$ over a single tree.

Using these simplifications, equation (12) becomes:

$$f_i = \int_\Omega C D(p) \|\mathbf{v}(t)\| \mathbf{v}(t).\phi_i(p) dp \qquad (14)$$

$$= C \|\mathbf{v}(t)\| \mathbf{v}(t). \int_\Omega D(p) \phi_i(p) dp \qquad (15)$$

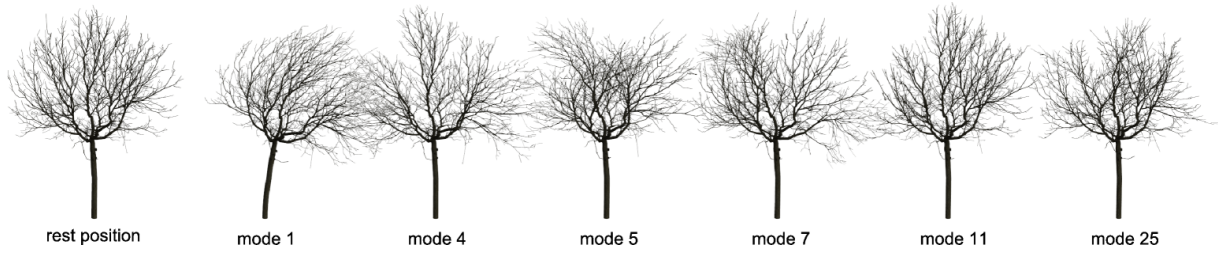It results that if we precompute the wind projection vectors p_i defined as:

$$p_i = C \int_\Omega D(p) \phi_i(p) dp$$

then the modal wind load can be easily computed at runtime using:

$$f_i = \|\mathbf{v}(t)\| \mathbf{v}(t).p_i \qquad (16)$$

Figure 4: *The modes of deformation for the walnut model. During animation the final displacement of a tree is a combination of its modal deformations. The modes are sorted in ascending order of their natural frequencies.*

5. Implementation issues

Mechanical modeling, modal analysis and precomputation of the projection basis require the use of complex algorithms that must be performed using specialized tools. We used the software Cast3m 2000 [CAS] specialized in the study of mechanical structures using the finite element method.

5.1. GPU implementation

A powerful aspect of the modal representation is that most computations are completely independent. It thus allows to take full advantage of the massive parallel computing power offered by present graphics hardware (GPU). This enables us to animate large forest scenes in real time (see section 6).

The two main parameters of our tree model are the number of nodes of the skeleton and the number of modes kept for animation.

1. As can be seen in equation (10) the updated position of each node can be computed independently from the others using only the dynamic modal state (q_i). Thus the nodes positions can be stored in a texture in GPU memory (matrix $\mathcal{U}$ of figure 5) and updated using a fragment program and an offscreen buffer. Animating the tree mesh can then be done by reading this texture in a dedicated vertex program. Any level-of-detail rendering model using a (possibly time-varying) subset of the animation nodes can be used, without the need to update the remaining unused nodes.
2. Adjusting the number of modes used for the computation of updated nodes positions gives a handful tradeoff between animation quality and framerate. For distant views of forests only the first few modes needs to be kept, while adding more modes at close range adds detail to the animation. As for the nodes, each modal state ($q_i, \dot{q}_i$) can be updated independently from the others, they are stored in a texture in GPU memory (matrix $\mathcal{Q}$ of figure 5), and updated by a dedicated fragment shader.

The overall animation process can be summarized in three main steps (see figure 5).

1. Update of the modal state for each mode of each tree

(item 1 and 2 of section 3.3). It requires to compute the local wind speed for each tree instance.
2. Deformation of the skeletons from the rest position (item 3 of section 3.3)
3. Rendering each tree with deformed geometry using skinning on the animated skeleton.

5.2. Error correction

Modal deformation being a linear approximation of the displacement of the branches nodes, it is mostly adapted for low amplitude motion. Notably, branches lengths are not kept constant when submitted to strong wind. We cannot rely on using rotating branches (like in [Zio07]) as it would break parallelism between nodes computations and does not extend to complex structures. This classical issue of modal analysis could be addressed by modal warping [CK05]. In our specific case of tree animation we observed that when a single mode i is selected, the corrected displacement of each node n lies on a smooth trajectory (see figure 6) which can be faithfully approximated by a quadratic curve **u** after a non linear reparameterization s:

$$\mathbf{u}(q) = s \cdot \mathbf{v} + s^2 \cdot \mathbf{w} \quad \text{with} \quad s = a \arctan(b \cdot q) \quad (17)$$

Our error correction consists in replacing equation (10) by:

$$\mathbf{u}_n = \sum_i s_{i,n} \cdot \mathbf{v}_{i,n} + (s_{i,n})^2 \cdot \mathbf{w}_{i,n} \quad (18)$$

$$\text{with} \quad s_{i,n} = a_{i,n} \arctan(b_{i,n} \cdot q_i) \quad (19)$$

Vectors $\mathbf{v}_{i,n}$, $\mathbf{w}_{i,n}$ and scalars $a_{i,n}$, $b_{i,n}$ are stored in matrix Φ (figure 5) instead of the displacement vectors (roughly doubling its size). They are optimized using uniformly sampled deformations, corrected by pulling each node towards its parent to match initial branches lengths, in a depth-first fashion. The results show that this approach yields a sufficiently good correction for branches lengths to allow the application of a strong wind load.

6. Results

We tested our technique on various models on an Intel Xeon 3.2 GHz with a GeForce 8800 GTS. The following results
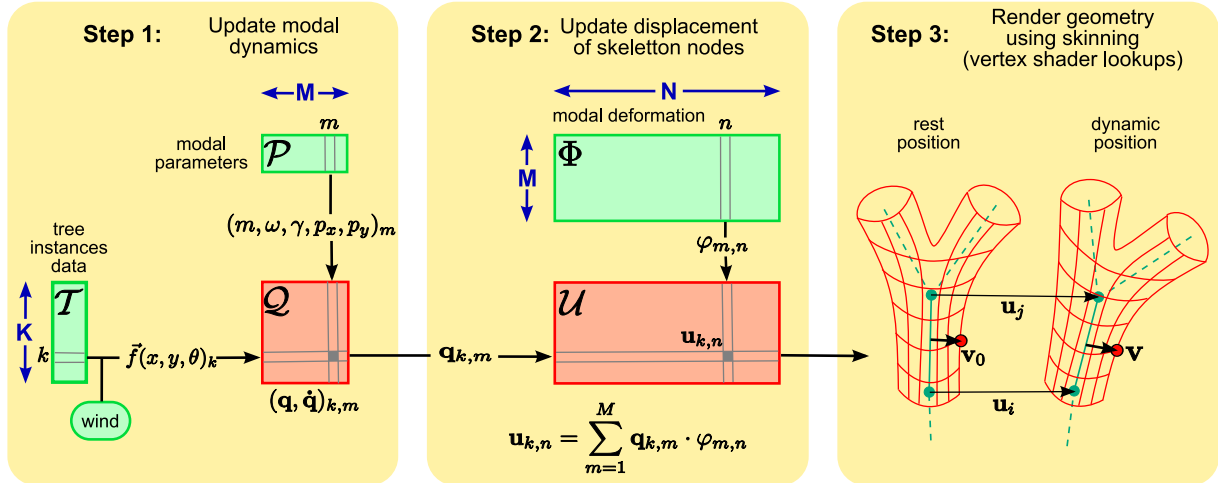
Figure 5: *Computation steps of the animation framework. K is the number of tree instances, M the number of modes used for the animation and N the number of control points of the skeleton. The colored rectangles represent textures stored in GPU memory. The input matrices $\mathcal{T}$, $\mathcal{P}$, Φ are constant while the matrices $\mathcal{Q}$, $\mathcal{U}$ store the dynamic variables. $\mathcal{T}$ contains the tree instance data (position, orientation); $\mathcal{P}$ contains the modal parameters used to compute the dynamics; Φ contains the modal deformations of the skeleton control points; $\mathcal{Q}$ contains the modal states $(q_i, \dot{q}_i)$; $\mathcal{U}$ contains the skeleton's node displacements. Several representations can be used for the wind such as a procedural equation or an animated flow field.*

Figure 6: *Correct trajectories obtained by selecting each mode successively (one color per mode). These trajectories can be faithfully approximated by second degree polynomial curves.*

Figure 7: *The oak tree model.*

were obtained on a walnut tree digitized from precise measurements on a real tree (from [SRG97], see figure 8), with 25 modes. The accompanying video involves this model and an oak model with 50 modes generated using [RLP07] (see figure 7). It demonstrates the resulting realism and some of the possibilities offered by the presented technique.

Our implementation can animate and render over 4000 trees while maintaining real-time framerates (30 Hz minimum). For the walnut tree with 3437 nodes, the undecimated version of the mesh is made of approximately 120000 vertices while the most decimated level of details only uses 300 vertices. Note that our LOD implementation is unsophisticated and could be much more optimized. The only significative data stored in GPU memory is the nodes displacements texture (array $\mathcal{U}$ in figure 5) whose size is $K \times N$ (*e.g.* 10 Mb for 1000 trees with 3000 control points each).

Tables 1 and 2 show that it is always possible to reach high framerates by making a tradeoff between the number of trees and the animation level of detail (*i.e.* the number of nodes and the number of modes).
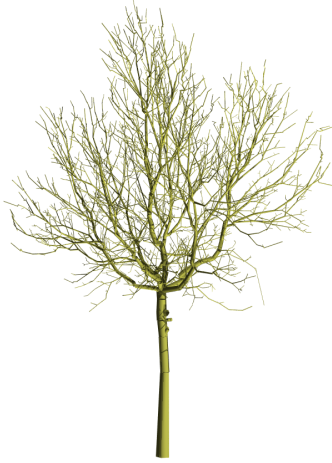
Figure 8: *Digitalized walnut tree.*

K \ N	20	100	500	1000	2000	8000
100				712	385	103
1000		690	169	87	43	11
8000	426	102	21	11		

Table 1: *InFluence of the number of trees (K) and the number of nodes (N) on animation framerates (in Hz), with 25 modes and rendering deactivated.*

M \ K	500	1000	2000	4000
5	73	39	19	10
15	57	29	14	7
25	43	22	11	6

Table 2: *Influence of the number of modes (M) and the number of trees (K) on the framerate (in Hz). The model has 3437 nodes and approximately twice as much vertices, and the LOD system is deactivated to ease interpretation (i.e. every node is always updated).*

7. Limitations and future work

As explained in section 3.3, the omission of the vibration modes with highest frequencies strongly reduces the computational cost of simulation but also removes the high frequency motion of tiny branches and leaves. However this can be replaced by using techniques such as [OTF*04] for close view and animated textures for distant views. In our implementation, the leaves rigidly follow the nodes they are attached to. But the complexity of branches animation already provides a very convincing motion.

Mathematically, the main limitation of our method is the assumption that the wind is spatially uniform over each tree.

In particular the attenuation of the wind by the tree is not taken into account (*i.e.* the branches in the 'back' of the tree should receive less wind). It does not lower visual realism as the modal dynamics keeps natural oscillatory behaviours. However precomputing a more expressive basis would allow to increase animation precision.

Finally the parallelism of the method coupled with efficient GPU programming enables the animation of thousands of trees. A more aggressive level-of-detail scheme would allow real-time rendering of even larger forest scenes.

8. Conclusion and acknowledgement

We showed a new approach to compute the modal projection of the wind load allowing a drastic reduction of computations at runtime. To this end, we introduce a precomputed basis for the projections of interactive directional wind in a modal animation framework.

Our implementation shows that it is possible to efficiently animate and render thousands of trees. We think the presented technique is perfectly adapted to real-time applications such as computer games or simulators.

Appendix

In the lineic representation of the tree, each element is defined in its local frame where the x-axis is in the direction of the segment at rest. The deformation of an element is the interpolation of its border nodes displacement: linear for the main axis and Hermitian for the y and z axis.
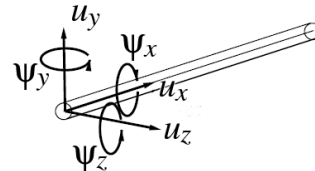
Figure 9: *The six degrees of freedom of a node (figure taken from [Sta97]).*

A node has thus six degrees of freedom (see fig. 9): the three spatial displacements (u_x, u_y, u_z), the torsion around the main axis (ψ_x) and the derivatives ψ_y and ψ_z defined by [GR94]:

$$\psi_y = -\frac{du_z}{dx} \quad \text{and} \quad \psi_z = \frac{du_y}{dx}$$

This interpolation is defined such that, at time t, the displacement $\delta_p(t)$ of any point $p \in \Omega^e$ is:

$$\delta_p(t) = N^e(p)\mathbf{u}^e(t) \tag{20}$$

where $\mathbf{u}^e(t)$ is the 12-dimensional vector containing the displacements of the two element's border nodes, and $N(p)$ the *shape function matrix* is:

$$N^e(p) = \begin{pmatrix} l_1 & . & . & . & . & . & l_2 & . & . & . & . & . \\ . & h_1 & . & . & . & h_2 & . & h_3 & . & . & . & h_4 \\ . & . & h_1 & . & h_2 & . & . & . & h_3 & . & h_4 & . \\ . & . & . & l_1 & . & . & . & . & . & l_2 & . & . \end{pmatrix}$$

where the dots represent zeros, l_1 and l_2 are the coefficients of a linear interpolation and the h_i are the cubic Hermite basis functions.

In accordance with most models of trees, branches are modeled as cylindrical beams [SFL06]. The local stiffness K^e and mass M^e matrix are defined as the discretization of the potential energy ν and kinetic energy τ of an element of length l [GR94]:

$$\begin{aligned} \nu &= \frac{1}{2} \int_0^l EA(\delta u_x)^2 + EI((\delta^2 u_y)^2 + (\delta^2 u_z)^2) + GJ(\delta \psi_x)^2 dx \\ &= \frac{1}{2} \mathbf{u}^{eT} K^e \mathbf{u}^e \end{aligned}$$

$$\begin{aligned} \tau &= \frac{1}{2} \int_0^l \rho A \left(\dot{u}_x^2 + \dot{u}_y^2 + \dot{u}_z^2 \right) + \rho J \, \dot{\psi}_x^2 dx \\ &= \frac{1}{2} \dot{\mathbf{u}}^{eT} M^e \dot{\mathbf{u}}^e \end{aligned}$$

Here the dot notation refers to time derivatives and $\delta = \frac{d}{dx}$.

In our implementation, we typically use the following values: $E = 10^{10} Pa$, $G = 2.6E$, $\rho = 10^3 kg.m^{-3}$. From the beam model used, we have $I = \pi r^4 / 16$, $J = \sqrt{2}I$, $A = \pi r^2$ where r is the average radius of the element.

These local matrices are then transformed to express the energies in function of the nodes displacements defined in the global frame of the tree. Finally, they are assembled together in the global matrices M and K of equation (1). A description of this procedure has been given by [Sta97] or can be found in specialized literature such as [GR94].

References

[AK06]   AKAGI Y., KITAJIMA K.: Computer animation of swaying trees based on physical simulation. *Computers & Graphics 30*, 4 (2006), 529–539.

[Bar96]   BARAFF D.: Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 137–146.

[CAS]   Cast3m 2000. http://www-cast3m.cea.fr/.

[CK05]   CHOI M. G., KO H.-S.: Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Transactions on Visualization and Computer Graphics 11*, 1 (2005), 91–101.

[dL08]   DE LANGRE E.: Effects of wind on plants. *Annual Review of Fluid Mechanics 40* (2008), 141–168.

[DRF06]   DIENER J., REVÉRET L., FIUME E.: Hierarchical retargetting of 2D motion fields to the animation of 3D plant models. In *Symposium on Computer Animation, SCA 06, September, 2006* (Sept. 2006), ACM-Siggraph/Eurographics, pp. 187–195.

[GCF01]   GIACOMO T. D., CAPO S., FAURE F.: An interactive forest. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)* (2001), Springer, pp. 65–74.

[GR94]   GÉRADIN M., RIXEN D.: *Mechanical Vibrations: Theory and Application to Structural Dynamics*. 1994.

[HSO03]   HAUSER K. K., SHEN C., O'BRIEN J. F.: Interactive deformation using modal analysis with constraints. In *Graphics Interface* (2003), pp. 247–256.

[JBP06]   JAMES D. L., BARBIČ J., PAI D. K.: Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (SIGGRAPH 2006) 25*, 3 (Aug. 2006).

[JF03]   JAMES D. L., FATAHALIAN K.: Precomputing interactive dynamic deformable scenes. *ACM Transactions on Graphics (SIGGRAPH 2003) 22*, 3 (July 2003), 879–887.

[JP02]   JAMES D. L., PAI D. K.: Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), pp. 582–585.

[JTCW07]   JAMES D. L., TWIGG C. D., COVE A., WANG R. Y.: Mesh ensemble motion graphs: Data-driven mesh animation with constraints. *ACM Transansaction Graphics 26*, 4 (2007).

[OTF*04]   OTA S., TAMURA M., FUJIMOTO T., MURAOKA K., CHIBA N.: A hybrid method for real-time animation of trees swaying in wind fields. *The Visual Computer 20* (2004), 613–623(11).

[PW89]   PENTLAND A., WILLIAMS J.: Good vibrations: modal dynamics for graphics and animation. *ACM Transactions on Graphics, Proceedings of the SIGGRAPH conference 23*, 3 (1989), 207–214.

[RdLM08]   RODRIGUEZ M., DE LANGRE E., MOULIA B.: A scaling law for the effects of architecture and allometry on tree vibration modes. *Accepted in American Journal of Botany* (2008).

[RLP07]   RUNIONS A., LANE B., PRUSINKIEWICZ P.: Modeling Trees with a Space Colonization Algorithm. Ebert D., Merillou S., (Eds.), Eurographics Association, pp. 63–70.

[SF92]   SHINYA M., FOURNIER A.: Stochastic motion-motion under the influence of wind. *Comput. Graph. Forum 11*, 3 (1992), 119–128.

[SFL06]   SELLIER D., FOURCAUD T., LAC P.: A finite element model for investigating effects of aerial architecture on tree oscillations. *Tree Physiology 26* (2006), 799–806.

[SO99]   SAKAGUCHI T., OHYA J.: Modeling and animation of botanical trees for interactive virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology (VRST '99)* (1999), pp. 139–146.

[SRG97]   SINOQUET H., RIVET P., GODIN C.: Assessment of the three-dimensional architecture of walnut trees using digitising. *Silva Fennica 31* (1997), 265–273.

[Sta97]   STAM J.: Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Comput. Graph. Forum 16*, 3 (1997), 159–164.

[Web08]   WEBER J. P.: Fast simulation of realistic trees. *IEEE Computer Graphics and Applications 28*, 3 (2008), 67–75.

[Zio07]   ZIOMA R.: *Gpu gems 3*. Addison-Wesley Professional, 2007, ch. Gpu-generated procedural wind animations for trees (chapter 6).

$\therefore$

5.5   CREATING AND ANIMATING SUBJECT-SPECIFIC ANATOMICAL MODELS

B. Gilles, L. RevÃ©ret, D.K. Pai Computer Graphics Forum, 29(8), dec 2010.

Creating and animating subject-specific anatomical models

B. Gilles [1], L. Revéret[2] and D. K. Pai [1]

[1] Department of Computer Science, University of British Columbia, Canada
[2] EVASION, INRIA Rhône-Alpes, France

Abstract

Creating and animating subject-specific anatomical models is traditionally a difficult process involving medical image segmentation, geometric corrections and the manual definition of kinematic parameters. In this paper, we introduce a novel template morphing algorithm that facilitates 3D modeling and parameterization of skeletons. Target data can be either medical images or surfaces of the whole skeleton. We incorporate prior knowledge about bone shape, the feasible skeleton pose, and the morphological variability in the population. This allows for noise reduction, bone separation, and the transfer, from the template, of anatomical and kinematical information not present in the input data. Our approach treats both local and global deformations in successive regularization steps: smooth elastic deformations are represented by an displacement field between the reference and current configuration of the template, while global and discontinuous displacements are estimated through a projection onto a statistical shape model and a new joint pose optimization scheme with joint limits.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

Biomechanics-based animation offers an exciting degree of realism as shown in recent studies [KM04,TSB*05,SKP08]. These methods require, however, accurate geometric models of bones and soft-tissues (muscles, skin, ligaments and cartilages). Their creation is usually time-consuming and needs to be performed independently for each subject, making such approaches impractical for graphics applications. Although there are semi-automatic methods to extract bones from medical images (e.g., thresholding of Computed Tomography images) [KM04], a significant manual work is always still required: bone separation, labeling, definition of the animation skeleton, soft-tissues segmentation and geometric corrections . In this paper, we tackle this problem through a new co-registration method that can automatically align generic skeletons to multi-modal data (surfaces and images). The key idea is to mix local deformations and a global registration to recover both the morphology and the pose of the target skeleton. Our new regularization method combines shape, statistical and joint limits constraints, and allows us to treat noisy and low resolution models and images. We are

also able to transfer features from the template (geometric details, animations) that were not necessarily acquired with the considered modality.

Registration aims at finding spatial correspondences between datasets. In other words, the goal is to find a deformation field that aligns a template to a target dataset. It is a central problem in the computer graphics and image processing communities [AFP00, ZF03]. The main difficulties are to find an adequate similarity measure that is as-convex-as-possible and a good parameterization of the deformation through the introduction of prior information. Registration has been mainly studied in the context of rigid alignment and small deformations. The musculoskeletal system however presents a large geometric variability in terms of morphology and pose. It makes registration difficult by increasing the number of local minima.

Our method addresses these issues by performing a robust co-registration of bone surfaces through constrained local and global deformations. Constraints are applied by regularizing an initial correspondence vector field between the datasets (Section 3.2). . Smooth elastic deformations, ac-
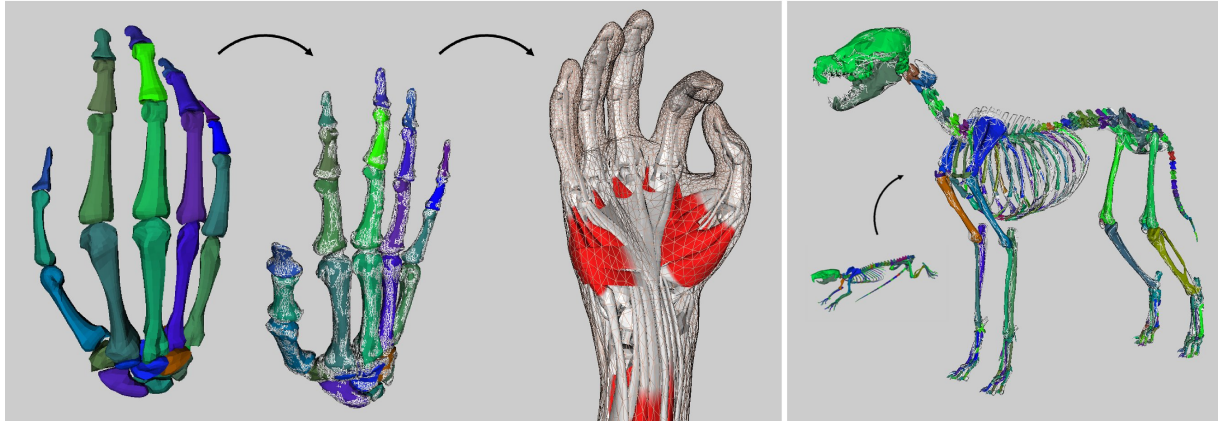
Figure 1: *Our method automatically warps template skeleton models to subject-specific anatomy and posture (left figure: hand registration; right figure: rat/dog). This allows transferring generic geometric details and animations (middle figure: transfer of hand soft-tissues and posture)*

counting for morphological differences, are represented by an displacement field between the reference and current configuration of the template (Section 3.3). The global evolution is ensured by a projection to a statistical shape model (Section 3.4) and a new joint pose optimization scheme with joint limits (Section 3.5). Finally, in Section 4, we illustrate the benefits of our methods through the automatic registration of a human hand model and a rat model to manually segmented surfaces and to Magnetic Resonance Images (MRI). As a result, we obtain subject-specific models, with smooth separated bones, that can be directly animated. We show how geometric and kinematic information can be mapped from the template to the target.

2. Related work

Registration consists in iterating two basic steps: first, the distance between the current position of the template and the target is estimated; secondly, this distance is minimized by deforming the template [AFP00, ZF03]. One class of methods, called *variational methods*, globally evolves the model along its degrees of freedom in order to minimize the distance. Alternatively, *pair-and-smooth* approaches locally minimize the distance through correspondence computations, and then regularize those displacements to satisfy deformation constraints. This approach is more suited in our context since our system has a high number of coupled degrees of freedom. It allows treating shape and inter-object constraints in successive regularization steps.

Correspondence computation: In discrete models, correspondences are defined for each vertex as the displacement that maximizes a certain similarity measure. Extrinsic similarity measures, based on the current configuration of the

surfaces in the Euclidean space, are widely used for simplicity [BBK09]. The popular Iterative Closest Point algorithm (ICP) results from the computation of closest points between the template and target surfaces [BM92]. In image registration, pairing is performed by locally maximizing the image similarity between the template image and target image. Several correlation measures have been proposed for a range of imaging modalities [ZF03]. Local pairing is not robust for large displacements however, unless there are few degrees of freedom and a clear global minimum. Robust correspondences can be better achieved by computing similarity over the entire spatial domain. In surface registration, researchers have considered the distance between rotation and translation invariant local shape descriptors, built from differential geometric quantities [HAWG08] . Similarly, features can be extracted from images based on the local intensity distribution [ZF03], such as the histogram moments [She07]. Contrary to closest point methods, feature correspondence needs propagation and smoothing steps to ensure a local consistency of the alignement [HAWG08]. Optimization of the pairing stage has been studied through voting techniques that minimize distortions of the template after registration [LF09, ZSCO*08]. Intrinsic properties of shapes, such as geodesic distances, are interesting because they are quasi-invariant under object pose and current deformation. Shape embedding techniques have been developed to enhance such properties, such as in spectral embedding methods [MHK*08], conformal mapping [LF09] or the medial axis transform [SSGD03]. Intrinsic methods attempt to find common parameterizations between template and target surfaces. They are global but sensitive to topological noise [BBK09].

As-rigid-as-possible deformation: Most surface and image

registration techniques assume a smoothly varying motion field over the spatial domain to avoid distorting the template excessively [ZF03]. Hence, regularization techniques typically try to enforce . The simplest way is to perform rigid registration to recover the dominant motion. The iterative closest point (ICP) algorithm [BM92] is a widely used approach that finds, at each iteration, the best rigid transform approximating the current target displacements. In the generalized gradient approach [CMP*07], target displacements are filtered to remove non-isometric components from the deformation field. In computer graphics, various techniques have been developed to generate as-rigid-as-possible deformations in order to mimic elasticity. Sorkine et al. [SA07] iteratively minimize a energy. Müller et al. [MHTG05] blend closest rigid transforms in their *shape matching* framework.

While the assumption of smooth displacements is acceptable for a single object, it is inaccurate at boundaries when the relative displacement between objects is large, which is the case for bones (see Fig. 2). In this context, researchers have tried to design piecewise quasi-isometric displacements fields. Arsigny et al. [ACPA06] introduced polyrigid and polyaffine transformations for image registration. Wang et al. [WHQ05] proposed a spline-based deformation technique that incorporates rigid components. For registering the skeletal system, articulated rigid motion has been considered in [KM04, STC*03, PDD*05]. Contrary to us they do not handle joint limits, relative translations and cyclic skeletal structures, and do not perform a simultaneous non-rigid registration. In motion capture, the underlying pose of the animation skeleton is computed from fiducial marker correspondences. Energy minimization approaches have been developed such as in [ACP02]. In [OBBH00], O'Brien et al. compute joint centers from markers motion (i.e., they register an acyclic chain of scalable rigid bodies). To smoothly deform surrounding soft-tissues from the articulated motion, numerous skinning techniques have been proposed in the graphics community. For skin registration, skeletal subspace deformation (or linear blend skinning) with an automatic tuning of influence weights has been presented in [VBMP08, HAWG08, CZ09].

Simulation of articulated rigid bodies: The fast simulation of constrained rigid bodies is an important and still open problem in computer graphics. Constrained dynamics approaches generate physically-based motion by solving the unknown constraint forces at contact or joint locations [Bar94]. Acyclic constraints between articulated bodies can be exactly and simultaneous enforced with a linear complexity in the number of joints either using generalized or maximal coordinates formulations [Bar96]. However, auxiliary constraints such a joint limits, contacts and loop closures require a specific treatment that is generally an iterative energy minimization process [Fau99, XWY*09]. The introduced inequality relations make the system difficult to solve numerically (i.e linear complementary problem). Besides global optimization methods, a simple local method
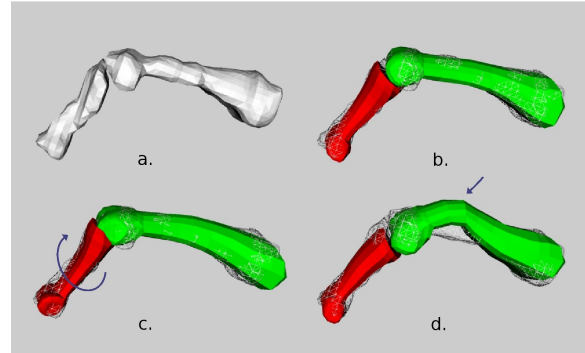
Figure 2: *Registration of a generic finger model to a manually segmented surface (a). The independent registration of the bones (c) and registration of the whole (d) present distortions and are sensitive to local minima (blue arrows), while our method (b) maintains realistic joint transforms and successfully decouples smooth and discontinuous deformations*

that treats all joints sequentially consists in applying spring penalty forces to enforce constraints [MW88, XWY*09], but suffers from stiffness and stability problems. Impulse-based methods [MC94, WTF06] solve this stability problem by altering the velocity of body pairs to guaranty a non violation of the constraint in the next frame. Another class of methods, into which our method falls, is the class of position-based methods. The goal is to procedurally adjust rigid body positions and orientations when constraints are not met [GC94, LBKH00]. Although not physically accurate, these static methods generate plausible results at a cheaper computational cost than constrained dynamics approaches.

3. Methods

3.1. Overview

The purpose of the algorithm is to find a deformation field that aligns a discrete surface model of a skeleton (the template) to a surface or 3-dimensional image (the target). Let $\mathbf{x^r}$ and $\mathbf{x}$ be the vertex position vectors of the template in the reference and deformed configurations. Figure 3 illustrates the different components of our registration framework. At each step of the process, correspondences $\mathbf{v}$ are estimated for each vertex (Section 3.2). For shape regularization, we approximate these correspondences with an as-rigid-as-possible displacement field $\tilde{\mathbf{v}}$, where the distance to rigid motion is locally minimized (Section 3.3). This elastic registration process is continuously iterated and vertices are deformed through $\mathbf{x} = \mathbf{x^r} + \tilde{\mathbf{v}}$ until convergence. With plasticity, shapes undergo permanent changes: we update the reference positions $\mathbf{x^r}$. To derive global deformations, coherent with shape variability in the population, we project $\mathbf{v}$ onto principal components of a statistical model (Section 3.4). Finally, we adapt the reference skeleton pose with joint limit
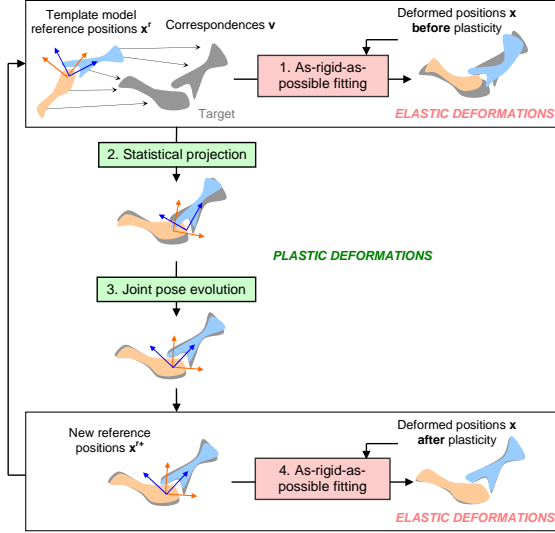
Figure 3: *Overview of our registration framework*

the reference image. It is the concatenation of moments at different orders ($n = 0, 1, 2$) and different resolutions ($s = s_0, 2.s_0, 4.s_0$) normalized between 0 and 1. During the registration, the similarity at the position ($\mathbf{x_i^r} + \mathbf{v}$) in the target image is simply given by $\Delta_i = \Pi_j(1 - |\tilde{a}_{i,j} - a(\mathbf{x_i^r} + \mathbf{v})_j|)$. We find the correspondence vector $\mathbf{v}$ by maximizing Δ_i in the neighborhood of the current position $\mathbf{x_i}$ via a local search along the normal direction of the surface. A threshold of 0.2 is applied to avoid maxima with a small Δ_i (see Fig. 4).
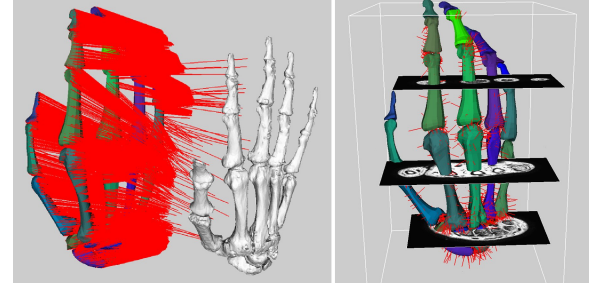
Figure 4: *Surface correspondences (left) vs. image correspondences (right). The target datasets are respectively a surface (in white) and an MRI volume (the bounding box and three sample slices are shown)*

constraints (Section 3.5). All these regularization steps allow incorporating shape, statistical and kinematic prior information.

3.2. Correspondence Computation

In the section, we present how we compute target (unconstrained) vertex displacements for template-to-surface and template-to-image registration.

Surface correspondences: Since features may not be clearly and uniquely identifiable in our datasets (e.g., fingers have quasi-similar shapes) and because of shape and topological noise (e.g., bones in close neighborhood), we apply a simple (extrinsic) closest point method: at each registration step, the deformed template (positions $\mathbf{x}$) is projected to the target surface and vice-versa. Target displacements $\mathbf{v_i}$ of the reference vertices $\mathbf{x^r}$ are obtained through a weighted sum of the resulting point-to-triangle vectors (see Fig. 4).

Image correspondences: Consider a reference volumetric image, where a manual registration of the generic model has been performed, and an image that we want to segment. The goal is to find the correspondence vector $\mathbf{v}$ that maximizes the image similarity in the neighborhood of each vertex. As a similarity measure, we compute the correlation of feature vectors based on histogram moments. They are rotation and translation invariant. In a spherical region of radius s around the position $\mathbf{p}$, the frequency of intensity i is noted $h(\mathbf{p_s}, i)$ and the number of voxels N. Contrary to [She07], we compute the moments around the mean intensity in order to combine gradient and intensity attributes. The central moment of order n is given by: $m(\mathbf{p_s}, n) = \sum_i (i - m(\mathbf{p_s}, 0)/N)^n h(\mathbf{p_s}, i)$. For each vertex i, we build a reference feature vector $\tilde{\mathbf{a}}_\mathbf{i}$ from

3.3. Deformation

To enforce shape constraints, that is to minimize distortions from the template shape, we simulate . We regularize the deformation field $\mathbf{v}$ through a local stiffening process. The particle influence weights are computed as the Voronoï surface of each vertex, and can be viewed as lumped particle masses m_i. This approach is referred as *shape matching* [MHTG05, RJ07, GP08]. Formally, let ζ_i be a cluster of cardinality $|\zeta_i|$ centered on the vertex i such as $\zeta_i = \{j : d(\mathbf{x_i^r}, \mathbf{x_j^r}) \leq S\}$ where S is the cluster size, and $d(,)$ the distance measure. For this cluster ζ_i, barycenters are noted $\bar{\mathbf{x}}_\mathbf{i}^\mathbf{r}$ and $\bar{\mathbf{v}}_\mathbf{i}$, and the optimal rotation $\tilde{\mathbf{R}}_\mathbf{i}$ The final regularized deformation field is:

$$\tilde{\mathbf{v}}_\mathbf{i} = \sum_{j \in \zeta_i} \frac{\tilde{\mathbf{R}}_\mathbf{j}(\mathbf{x_i^r} - \bar{\mathbf{x}}_\mathbf{j}^\mathbf{r}) + \bar{\mathbf{x}}_\mathbf{j}^\mathbf{r} + \bar{\mathbf{v}}_\mathbf{j}}{|\zeta_i|} - \mathbf{x_i^r} \qquad (1)$$

Fast summation: The   uses weighted sums of positions and covariance matrices, and computational time can be improved by exploiting cluster overlapping as in [RJ07]. Consider a parent vertex k and a child vertex i. The Boolean differences between the two clusters are $\zeta_i^+ = \{j : j \in \zeta_i - (\zeta_k \cap \zeta_i)\}$ and $\zeta_i^- = \{j : j \in \zeta_k - (\zeta_k \cap \zeta_i)\}$. Then, summation of a field data $\mathbf{u}$, within the cluster i can be quickly performed if the two clusters are redundant, through: $\Sigma_{j \in \zeta_i} \mathbf{u_j} = \Sigma_{j \in \zeta_k} \mathbf{u_j} + \Sigma_{j \in \zeta_i^+} \mathbf{u_j} - \Sigma_{j \in \zeta_i^-} \mathbf{u_j}$.

Shape matching based on unstructured points: In [RJ07], the cluster-based technique is applied to regular lattices.

With lattices, Boolean differences and the optimal vertex ordering are systematic (summation for every clusters can be done in three global passes). Here, as in [GP08], we directly use unstructured model vertices, suppressing the need for warping target displacements to the lattice and interpolating model deformation from the lattice. With unstructured points, the fast summation must follow an ordered list so that $\Sigma_{j \in \zeta_k} \mathbf{u_j}$ is available for computing $\Sigma_{j \in \zeta_i} \mathbf{u_j}$. In a template pre-processing phase, we build such a list using a propagation-based approach that maximizes at each step the gain g_i^k of having k as a parent of i: $g_i^k = |\zeta_i| - |\zeta_i^-| - |\zeta_i^+| - 1 = 2|\zeta_k \cap \zeta_i| - |\zeta_k| - 1$. This list building criterion is more optimal but more computationally demanding than the one of [GP08]. The accompanying video shows an example of the cluster ordering process.

3.4. Statistical Constraints

A common approach to improve the robustness of a registration process is to add prior knowledge about the variability of shapes across the population [CT01, ASK*05]. Principal component analysis (PCA) is widely used to derive the main variations of parameters living in a linear space (such as vertex positions) [Jol86]. However, parameterizing the displacement field of the registration using only these principal variations is too restrictive. To create new shapes, excursions from the statistical model need to be realizable. In our framework, excursions from a reference shape are modeled through the deformation field presented in the previous section. To add smooth statistical constraints, we only need to evolve the reference positions $\mathbf{x^r}$ along the principal components (PCs) after the elastic registration has converged. This is effectively a plastic behavior. We generate the PCs using examples of bone shapes generated with our algorithm. Correspondences across models are known, so there is no need to identify features as done in most shape PCA-based methods. We remove the rigid transform component by registering bones from the different subjects into the same frame using the optimal rigid alignment method. We then apply the PCA to the resulting vertex positions of all bones simultaneously (thus, we account for their mutual correlation). During registration, we evolve the reference positions $\mathbf{x^r}$ by projecting $(\mathbf{x^r} + \mathbf{v})$ onto the PCs: we rigidly align each bone to its mean shape, perform projection for all bones simultaneously and finally transform bones back to their original frame.

Updating joint coordinate systems: Due to the introduced deformations, we need to interpolate the rigid motion of the coordinate systems attached to the bones. Consider a coordinate system k in its reference configuration, represented by the 1×3 position vector $_k\mathbf{t}$ and 3×3 rotation matrix $_k\mathbf{R}$. We solve for $\{_k\mathbf{t}^+, _k\mathbf{R}^+\}$ in the new configuration based on the new vertex positions $\mathbf{x^{r+}}$ of the considered bone.
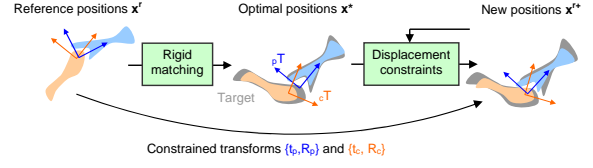

Figure 5:

3.5. Constrained Skeleton Pose Estimation

Because we do not account for displacement discontinuities, the dense elastic registration process is inaccurate around joints (see Fig. 2). To reduce these distortions, we adapt bone transforms in the reference configuration after convergence of the registration process. Following [HHN88], we compute, for each bone b, the optimal rigid transform $\{\mathbf{t_b^*}, \mathbf{R_b^*}\}$ approximating $\mathbf{v}$, . Because of noise and local minima however, unrealistic joint transforms can be achieved. Our goal is to instead find the valid bone transforms . Therefore, we solve it through an iterative treatment.

3.5.1. Joint Limits

Consider a joint between bones p and c. The associated local coordinates systems are represented by homogeneous matrices composed of translations and rotations: $_p\mathbf{T} = \{_p\mathbf{t}, _p\mathbf{R}\}$ and $_c\mathbf{T} = \{_c\mathbf{t}, _c\mathbf{R}\}$. The joint transform $\mathbf{J} = _p\mathbf{T}^{-1}{}_c\mathbf{T}$ can violate the anatomic joint limits. In a quaternion form, the joint transform is given by $\mathbf{J} = \{\mathbf{t}, q(\mathbf{r})\}$, where $\mathbf{t}$ represents the joint shift, $\mathbf{r} = \theta\mathbf{u}$ the rotation vector and $q(\mathbf{r}) = [\cos(\theta/2), \mathbf{u}\sin(\theta/2)]$ is the unit quaternion.

Angular limits: We choose to represent the space of feasible joint angles with an implicit function of $\mathbf{r}$. We parameterize this function using maximal angles in the six principal directions. When coordinate systems are appropriately chosen, they correspond to anatomical angles (e.g., flexion/extension, int/ext rotation, abduction/adduction, etc.). Joint ranges of motion can be measured and are well documented [PY05]. As an implicit surface, we construct an asymmetric ellipsoid passing through these six points, by combining one eighth of ellipsoids. This type of surface is smooth and allows simple inside/outside tests: let a, b and c be the three maximal angles corresponding to the one eighth of space containing $\mathbf{r}$, then angles are within the limits if $(r_x/a)^2 + (r_y/b)^2 + (r_z/c)^2 < 1$. When $\mathbf{r}$ is outside, we estimate the closest rotation by computing the closest point on the ellipsoid. This point is iteratively found using a simple Newton search. Note that we need to take the closest points from the two equivalent rotations $\mathbf{r} = \theta\mathbf{u}$ and $\mathbf{r'} = -(\pi+\theta)\mathbf{u}$. The space of rotation $\mathbf{r}$ vectors instead of the space of quaternion vectors $\mathbf{u}$ leads to a more accurate estimation of the distance between rotations [HUF05].

Shift limits: To enforce joint limits in terms of translation, we consider an identical limit t_M in the six directions (sphere

instead of ellipsoid), leading to a simpler formulation for inside/outside test and projection: $\|\mathbf{t}\|/t_M < 1$.

3.5.2. Displacement constraints

Projecting joint translations and rotations simultaneously and applying corrections to reach this desired state can lead to severe excursion from the optimal positions $\mathbf{x_i^*}$. To couple corrections in rotation and translation, we perform them sequentially while minimizing the quadratic error *err* at each step:

- **Constrained minimization for translations:** if the joint shift is invalid ($\|\mathbf{_pt} - \mathbf{_ct}\| > t_M$), . We have:

$$
\begin{aligned}
\mathbf{t_p} &= \frac{M_c}{M_c + M_p}\left(1 - \frac{t_M}{\|\mathbf{_pt} - \mathbf{_ct}\|}\right)(\mathbf{_pt} - \mathbf{_ct}) \\
\mathbf{t_c} &= \frac{M_p}{M_c + M_p}\left(1 - \frac{t_M}{\|\mathbf{_pt} - \mathbf{_ct}\|}\right)(\mathbf{_ct} - \mathbf{_pt})
\end{aligned}
\tag{2}
$$

- **Constrained minimization for rotations:** from the resulting translated positions, we solve for the optimal rotations of the two bones centered on $\mathbf{_pt}$ and $\mathbf{_ct}$ respectively. This maintains a constant joint shift and therefore does not break translation constraints. if the joint angle is outside the limits, we find the closest joint rotation using the method described in the previous section. From it, we update the position of one of the bone.
- **Global rigid transform:** we compute the optimal rigid transform of the two bones as a whole. In this step, the joint transform remains unchanged and constraints are met.

4. Results

4.1. Performances

We illustrate our methods with two generic anatomical models: a model of the rat skeleton, interactively built from micro CT data, and a model of the human hand, purchased from Snoswell Design. For all joints, we manually defined the local bone coordinate systems and joint limits in agreement with the literature and joint morphology. Limits in translations were slightly increased to cover the variability between subjects. In a pre-processing step, vertices were grouped into clusters subsequently optimized for fast summation, as described in Section 3.3. This was done for different cluster sizes, to control the during the registration.

Our algorithm has been implemented in C for testing. Where possible, it has been parallelized. All timings have been measured on a 2.4Ghz QuadCore machine, and exclude visualization time. We typically reach 90% of the CPU usage during the registration process. The gain in using fast summation depends on the cluster redundancy, and therefore on the object shape and cluster size. We experienced an average reduction of 85% of the number of summation values during each step of the deformation . To simulate deformation, our method achieves an average computation time of $2.5\mu s$ per vertex, per iteration ($100k$ vertices at 4fps). The

Generic Models	Rat	Hand
# Vertices	34817	7218
# Bones	214	27
# Joints	228	40
deformations	150*ms*	20*ms*
Skeleton pose estimation	40*ms*	7*ms*
PCA projection	-	7*ms*
Geometric correspondences # vertices of the target mesh	3000*ms* 50000	500*ms* 20000
Image correspondences	-	350*ms*

Table 1: *Summary of computational time per iteration, for each step of the registration process*

complexity of our joint pose optimisation scheme (Section 3.5) depends on the number of vertices (pre-compuation of the sums and covariance matrices), on the number of joints and on the number of loops in which all joints are treated sequentially. When all joints are in a valid state, the algorithm stops, so complexity also depends on the validity of the current skeleton pose. We summarize the average computation times for the two models in Table 1.

4.2. Surface Registration

Our input target surfaces were segmented in medical images. We manually delineated hand bones in MRI (resolution of $0.3 \times 0.3 \times 1mm$). It resulted in aliased surfaces and inaccurate interfaces (see Fig. 2 and Fig. 4). Rat skeletons were obtained by thresholding CT data (resolution of $25 \times 25 \times 25\mu m$). With this technique, the internal and external surfaces of the hard bone tissue are extracted, though only the external part is desired. We converted segmented images into meshes using a standard iso-surfacing pipeline (marching cubes, smoothing and mesh simplification).

Based on the correspondences to these surfaces, we applied our registration method. Starting from a complete rigid registration, more degrees of freedom are successively added by decreasing the (i.e., the cluster size). When the is under a small threshold , projection to the statistical model and joint pose estimation are performed to update the reference particle positions (plastic deformation). For the *hand* example shown in the Video, 8 samples are used for PCA. For the *rat* example, we did not have enough samples, and so used a fictitious scaling component which reflects the first principal component of a realistic statistical model.

Registration results typically obtained for the rat and the hand are illustrated in Fig. 1 and Fig. 6. Our method successfully corrects the by introducing prior geometric information for each bone, and prior functional information for each joint. It allows separating the different bones that where initially segmented as a whole. We also have an extreme example: the registration of the rat to a CG model of a dog (see Fig. 1 and Fig. 8). To handle the large morphological dif-

ferences and get correct correspondences, we segmented the dog skeleton and associated each target surface to the corresponding generic bones.
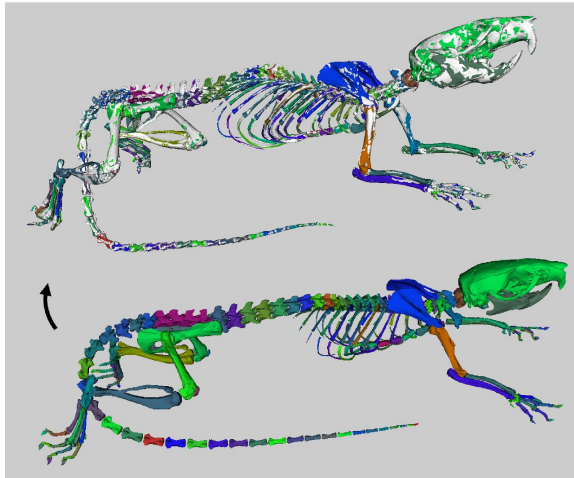
Figure 6: *Automatic registration of a generic rat model (bottom) to a surface extracted from CT data (in white)*

4.3. Image Registration

While CT thresholding is simple, the creation of target surfaces from MRI still requires a significant manual work. Here we apply the same methodology for image registration, in order to automatically segment bones in MRI. From one dataset, we obtain generic feature vectors through manual segmentation and surface registration as described previously. These vectors are used for registration in other datasets acquired with the same imaging protocol. For correspondence finding in our *hand* example, we used 20 samples with a decreasing search space from 20*mm* to 5*mm* inside and outside the surface, along the normal direction (see Fig. 4).

We validated our technique by checking the distance between the resulting models and manually segmented models. On average, we achieved an average distance of 0.8*mm* and the convergence time was approximately 3*min* (see Video). As soon as models are reconstructed for a subject, our algorithm can estimate skeleton poses in other datasets with the constraint of keeping bones rigid as shown in the Video and Fig. 7. In this case, we only have one cluster size set to infinite.

4.4.

4.5. Applications

Geometry transfer: After bone registration, one direct application is to enrich the subject specific models with details from the template such as surrounding soft-tissues (see
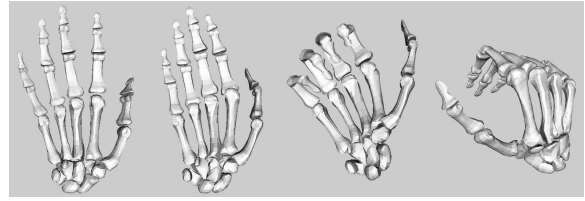
Figure 7: *Our algorithm can automatically track hand poses from low resolution MRI stacks*

Video and Fig. 1). Our deformation scheme can do that automatically: first, we preprocess generic clusters based on bone and soft-tissue vertices together. We transform bone vertices to their subject specific positions, constrain them to remain fixed and finally run the simulation to deform soft-tissues. As for registration, a faster converge is achieved by progressively decreasing .

Animation transfer: Bone local coordinate systems contain other important geometric details that are automatically warped (and that are usually time-consuming to set). Our method allows for transferring animations of the generic model using forward kinematics. We first make skeletons acyclic by breaking auxiliary constraints. Then, we simply propagate joint angles and shifts from the root bone to the leaves. Our registration algorithm keeps coordinate systems consistent with the geometry. This removes the usual ambiguity in the internal/external rotation angle, present in the registration of animation skeleton. Our method allows comparing animation from different subject and even different species in the same kinematics space (see Fig. 8). In our examples generic animations were obtained from biomechanical simulations and motion capture.
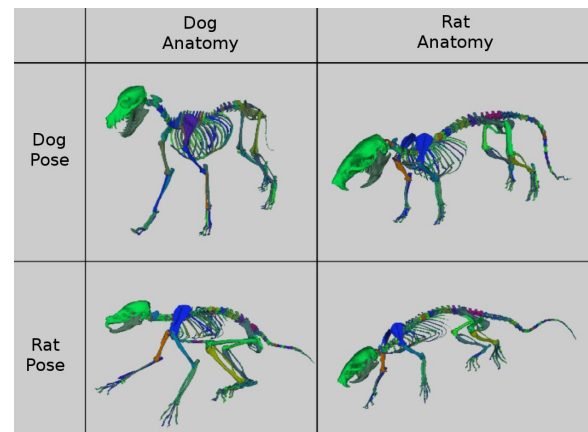
Figure 8: *Animation transfer from registered rat and dog skeletons*

Shape analysis and synthesis: By performing PCA on all

bones simultaneously we take into account their mutual correlation. The influence of the skeleton pose has been filtered out by rigidly registering individual bones. So the principal components only reflect variations of the morphology. With our method, we have built a statistical model based on 9 hands. The Video shows the first 4 modes. The hand is kept in the same pose using forward kinematic, as previously described. New synthetic shapes can be interpolated/extrapolated by adding to the mean a linear combination of principal components. Exaggerating normal anatomy is particularly relevant for modeling and animating cartoon-like characters (see Video).

5. Conclusion

Benefits: Our new method aligns generic skeleton models with subject-specific data by treating both local and global deformations. We have successfully incorporated prior knowledge about the shape, the feasible skeleton pose, and the variability in the population. Input data can be either surfaces of the skeleton as a whole, or low resolution medical images. Adding prior knowledge allows for noise reduction, bone separation and estimation of extra anatomical features through geometry transfer. This adds flexibility at the acquisition stage in terms of resolution and tissue specificity. Our method can be applied in geometric modeling and animation (morphing of animated biomechanical models) but also in the medical field (functional anatomy and comparative anatomy domains).

Limitations and future work: Our results are encouraging, however we would like to investigate validation more thoroughly. For soft-tissue geometry transfer, we will quantify the accuracy of using dense deformations only and check the amount of displacement discontinuity (e.g., sliding between surfaces, changes in attachment locations and tendon network topology). In this paper, we did not focus on the correspondence computation step. This is currently the main bottleneck in terms of complexity (95% of the whole computation time). This could be clearly improved by updating correspondences at each iteration instead of re-computing them. Using extrinsic similarity is also responsible of possible falls into local minima (e.g., , and rat tail not reaching the tip as shown in the Video). To improve this, a possible solution would be to mix feature correspondence methods that are more , global and robust [HAWG08, ZSCO*08, LF09], but more sensitive to noise in the input data. In future, we expect to use registration for parameterizing functional models of the musculoskeletal system, by fusing data from complementary modalities (e.g., merge information from motion capture, CT, MRI and histological cross-sections).

Acknowledgments

References

[ACP02] ALLEN B., CURLESS B., POPOVIC Z.: Articulated body deformation from range scan data. *ACM Transactions on Graphics 21*, 3 (2002), 612–619.

[ACPA06] ARSIGNY V., COMMOWICK O., PENNEC X., AYACHE N.: A log-Euclidean polyaffine framework for locally rigid or affine registration. *proc. of WBIR* (2006), 120–127.

[AFP00] AUDETTE M., FERRIE F., PETERS T.: An algorithmic overview of surface registration techniques for medical imaging. *Medical Image Analysis 4*, 3 (2000), 201–217.

[AHS03] ALBRECHT I., HABER J., SEIDEL H.-P.: Construction and animation of anatomically based human hand models. *Proc. of Symp. on Computer Animation '03 368* (2003), 98–109.

[ASK*05] ANGUELOV D., SRINIVASAN P., KOLLER D., THRUN S., RODGERS J., DAVIS. J.: Scape: Shape completion and animation of people. *Proc. of SIGGRAPH, ACM Trans. Graph.* (2005).

[Bar94] BARAFF D.: Fast contact force computation for nonpenetrating rigid bodies. *Proc. of SIGGRAPH* (1994), 23–34.

[Bar96] BARAFF D.: Linear-time dynamics using lagrange multipliers. *Proc. of SIGGRAPH* (1996), 137–146.

[BBK09] BRONSTEIN A., BRONSTEIN M., KIMMEL R.: Topology-invariant similarity of nonrigid shapes. *Int. J. Comp. Vision (IJCV) 81*, 3 (2009), 281–301.

[BM92] BESL P., McKAY N.: A method for registration of 3-d shapes. *IEEE Trans. PAMI 14*, 2 (1992), 239–256.

[CMP*07] CHARPIAT G., MAUREL P., PONS J., KERIVEN R., FAUGERAS O.: Generalized gradients: Priors on minimization flows. *Int. J. of Computer Vision 73*, 3 (2007), 325–344.

[CT01] COOTES T., TAYLOR C.: Statistical models of appearance for medical image analysis and computer vision, in medical imaging. *Proc. of SPIE 4322* (2001), 238–248.

[CZ08] CHANG W., ZWICKER M.: Automatic registration for articulated shapes. *Proc. of SGP, Comp. Graph. Forum* (2008).

[CZ09] CHANG W., ZWICKER M.: Range scan registration using reduced deformable models. *Proc. of Eurographics 28*, 2 (2009), 447–456.

[dAST*08] DE AGUIAR E., STOLL C., THEOBALT C., AHMED N., SEIDEL H.-P., THRUN S.: Performance capture from sparse multi-view video. *Proc. of SIGGRAPH, ACM Trans. Graph. 27*, 3 (2008).

[Fau99] FAURE F.: Fast iterative refinement of articulated solid dynamics. *IEEE TVCG 5*, 3 (1999), 268–276.

[GC94] GASCUEL J.-D., CANI M.-P.: Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer 10*, 4 (1994), 191–204.

[GP08] GILLES B., PAI D.: Fast musculoskeletal registration based on shape matching. *Proc. of MICCAI'08 2* (2008), 822–829.

[HAWG08] HUANG Q., ADAMS B., WICKE M., GUIBAS L.: Non-rigid registration under isometric deformations. *Comput. Graph. Forum 27*, 5 (2008), 1449–1457.

[HHN88] HORN B., HILDEN H., NEGAHDARIPOUR S.: Closed-form solution of absolute orientation using orthonormal matrices. *J. of the Optical Society of America 5* (1988), 1127–1135.

[HUF05] HERDA L., URTASUN R., FUA P.: Hierarchical implicit surface joint limits for human body tracking. *Computer Vision and Image Understanding 99*, 2 (2005), 189–209.

[Jol86] JOLLIFE T.: Principle component analysis. *SpringerVerlag, New York* (1986).

[KM04] KURIHARA T., MIYATA N.: Modeling deformable human hands from medical images. *Proc. of Symposium on Computer Animation'04* (2004), 355–363.

[LAGP09] LI H., ADAMS B., GUIBAS L. J., PAULY M.: Robust single-view geometry and motion reconstruction. *Proc. of SIGGRAPH Asia, ACM Trans. Graph.* (2009).

[LBKH00] LEE J., BAEK N., KIM D., HAHN J.: A procedural approach to solving constraints of articulated bodies. *Proc. of Eurographics, Short Paper* (2000).

[LF09] LIPMAN Y., FUNKHOUSER T.: Mobius voting for surface correspondence. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3 (2009).

[LSP] LI H., SUMNER R. W., PAULY M.: Global correspondence optimization for non-rigid registration of depth scans. *Proc. of SGP, Comp. Graph. Forum 27*, 5, 2008.

[MC94] MIRTICH B., CANNY J.: Impulse-based dynamic simulation. *The Algorithmic Foundations of Robotics* (1994).

[MHK*08] MATEUS D., HORAUD R., KNOSSOW D., CUZZOLIN F., BOYER E.: Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. *Proc. of IEEE CVPR* (2008), 1–8.

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. (Proc. of SIGGRAPH)* (2005), 471–478.

[MW88] MOORE M., WILHELMS J.: Collision detection and response for computer animation. *Proc. of SIGGRAPH 22*, 4 (1988), 289–298.

[OBBH00] O'BRIEN J., BODENHEIMER B., BROSTOW G., HODGINS J.: Automatic joint parameter estimation from magnetic motion capture data. *Proc. of Graphics Interface '00* (2000), 53–60.

[PDD*05] PAPADEMETRIS X., DIONE D., DOBRUCKI L., STAIB L., SINUSAS A.: Articulated rigid registration for serial lower-limb mouse imaging. *Proc. of MICCAI 3750* (2005), 919–926.

[PY05] PAPAIOANNOU G., YENI Y.: Biomechanics of joints. *Wiley Encyclopedia of Biomedical Engineering* (2005).

[RJ07] RIVERS A., JAMES D.: Fastlsm: Fast lattice shape matching for robust real-time deformation. *ACM Trans. Graph. (Proc. of SIGGRAPH) 26*, 3 (2007).

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. *Proc. of SGP* (2007), 109–116.

[She07] SHEN D.: Image registration by local histogram matching. *Pattern Recognition 40*, 4 (2007), 1161–1172.

[SKP08] SUEDA S., KAUFMAN A., PAI D.: Musculotendon simulation for hand animation. *Proc. of SIGGRAPH'08, ACM Trans. Graph. 27*, 3 (2008), 83.

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. *ACM Trans. Graph. (Proc. of SIGGRAPH) 25*, 3 (2006), 533–540.

[SSGD03] SUNDAR H., SILVER D., GAGVANI N., DICKINSON S.: Skeleton based shape matching and retrieval,. *Proc. IEEE Conf. on Shape Modeling and Applications* (2003), 130–142.

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. *Proc. of SIGGRAPH, ACM Trans. Graph.* (2007).

[STC*03] SEBASTIAN T., TEK H., CRISCO J., WOLFE S., KIMIA B.: Segmentation of carpal bones from ct images using skeletally coupled deformable models. *Medical Image Analysis 7*, 1 (2003), 21–45.

[TSB*05] TERAN J., SIFAKIS E., BLEMKER S., HING V. N. T., LAU C., FEDKIW R.: Creating and simulating skeletal muscle from the visible human data set. *IEEE TVCG 11* (2005), 317–328.

[VBMP08] VLASIC D., BARAN I., MATUSIK W., POPOVIC J.: Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics 27*, 3 (2008), 97:1–97:9.

[WHQ05] WANG K., HE Y., QIN H.: Incorporating rigid structures in non-rigid registration using triangular b-splines. *VLSM* (2005), 235–246.

[WTF06] WEINSTEIN R., TERAN J., FEDKIW R.: Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE TVCG 12* (2006), 365–374.

[XWY*09] XU W., WANG J., YIN K., ZHOU K., VAN DE PANNE N., CHEN F., GUO B.: Joint-aware manipulation of deformable models. *ACM Trans. Graph. (proc. of SIGGRAPH) 28*, 3 (2009).

[ZF03] ZITOVÁ B., FLUSSER J.: Image registration methods: a survey. *Image Vision Comput. 21*, 11 (2003), 977–1000.

[ZSCO*08] ZHANG H., SHEFFER A., COHEN-OR D., ZHOU Q., VAN KAICK O., TAGLIASACCHI A.: Deformation-driven shape correspondence. *Comp. Graph. Forum (Proc. of SGP) 27*, 5 (2008), 1431–1439.

$\therefore$

5.6 NATURAL CHARACTER POSING FROM A LARGE DATABASE

Xiaomao Wu, Maxime Tournier, Lionel Reveret IEEE Computer Graphics and Applications, 31(3), may 2011.

Natural Character Posing from a Large Motion Database

Xiaomao Wu ■ *Crytek*

Maxime Tournier and Lionel Reveret ■ *INRIA*

The game and movie industries have widely adopted motion-capture-based techniques to bring characters to life. Because the captured data is relatively expensive and might not match user requirements, researchers have intensively investigated how to reuse it in different scenarios.

Inverse kinematics (IK) is one of the most important and fundamental techniques for editing motion data, especially keyframe-based editing[1] and postprocessing such as foot-skate cleaning and floor penetration prevention. (*Footskate* refers to the feet skating on the ground during an animation playback. For more on IK, see the related sidebar.) Traditional optimization-based IK techniques have seen wide adoption. These techniques provide real-time speed; however, performance drops quickly when the joint limits are integrated. In such cases, we must solve a highly nonlinear constrained optimization problem. However, even if the joint limits are integrated, we might still obtain unnatural poses because joint limits won't guarantee a natural pose. Joint limits provide no information about the most likely pose humans in motion would use to reach given IK-handle positions. (In motion-editing applications, users employ IK handles to manipulate poses.)

One novel IK solution is *style-based IK*,[2] which is useful for generating poses with a specific style similar to that of an existing motion clip. In spite of style-based IK's robustness and efficiency, it can generate natural poses only in a narrow, human-reachable space because of its limited training capacity. (A human-reachable space is a space that human end-effectors—head, hands, and feet—can reach with the root joint being fixed.) Although you

could use more training data, the poses of a motion with several hundreds of frames might span only a narrow space in the entire reachable space (see Figure 1). Style-based IK also has difficulty training a model with more than a thousand frames, and the synthesizing speed becomes very slow.

Our NAT-IK system lets users interactively generate natural IK poses in a large human-reachable space. With NAT-IK, users need to train the model only once. The trained model parameters contain 30 Mbytes of data, which can be imported conveniently into existing animation packages. This data can be further compressed.

The Basis for NAT-IK

We based NAT-IK on three main observations. First, discrete poses, instead of continuous poses selected from a large motion database, are sufficient to create a robust IK algorithm. This algorithm enables us to train models from a large motion database.

Second, among all the methods for fast Gaussian processes (GPs), the best choice for IK problems is the *fully independent training conditional* (FITC) approximation.[3,4] Selecting a proper approximation method for GP is difficult. We've evaluated the available fast GP methods for IK problems. To the best of our knowledge, no one has reported on fast GP methods for interactive character animation. We've also evaluated the reconstruction error and the training and synthesizing speeds for different sparse methods.

An interactive inverse-kinematics approach robustly generates natural poses in a large human-reachable space. It employs adaptive *k*d clustering to select a representative frame set from a large motion database and employs sparse approximation to accelerate training and posing. Model training is required only once.

Related Work on Inverse Kinematics

Inverse kinematics (IK) has been well studied by researchers in mechanics and computer graphics. Michael Gleicher proposed a space-time motion-editing scheme that uses constrained optimization to solve the involved IK problems.[1] Jehee Lee and Sung Yong Shin proposed IK algorithms that combine analytical methods and optimization procedures.[2]

Douglas Wiley and James Hahn[3] and Charles Rose III and his colleagues[4] provided interpolation-based IK solutions. Their methods are robust because they produce synthesized poses by blending existing examples. Style-based IK, which Keith Grochow and his colleagues proposed, can interactively generate a pose similar to that of the training motion by learning a *scaled Gaussian process latent variable model* from motion data.[5] Both interpolation-based IK and style-based IK are limited to specific training motions. Users should carefully select motions that can meet their requirements and should ensure these motions can be blended correctly. This situation creates difficulties for

users. We aim to provide a general solution that requires minimal user intervention.

References

1. M. Gleicher, "Motion Editing with Spacetime Constraints," *Proc. 1997 Symp. Interactive 3D Graphics* (I3D 97), ACM Press, 1997, pp. 139–148, 195.
2. J. Lee and S.Y. Shin, "A Hierarchical Approach to Interactive Motion Editing for Human-Like Figures," *Proc. Siggraph*, ACM Press, 1999, pp. 39–48.
3. D.J. Wiley and J.K. Hahn, "Interpolation Synthesis of Articulated Figure Motion," *IEEE Computer Graphics and Applications*, vol. 17, no. 6, 1997, pp. 39–45.
4. C.F. Rose III, P.-P.J. Sloan, and M.F. Cohen, "Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation," *Computer Graphics Forum*, vol. 20, no. 3, 2001, pp. 239–250.
5. K. Grochow et al., "Style-Based Inverse Kinematics," *ACM Trans. Graphics*, vol. 23, no. 3, 2004, pp. 522–531.
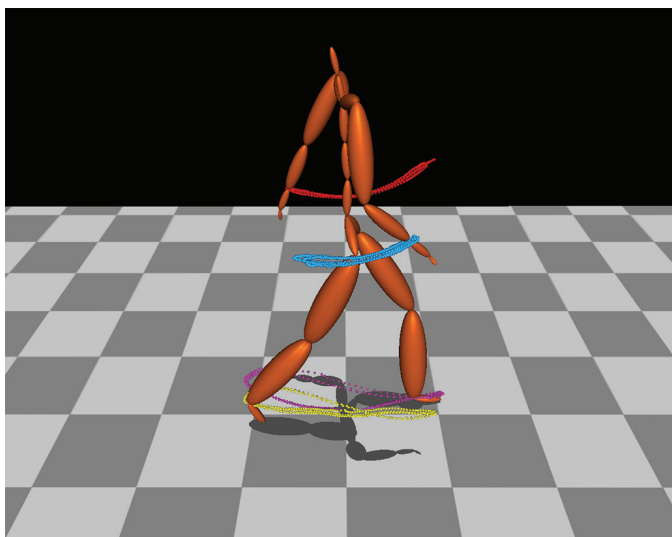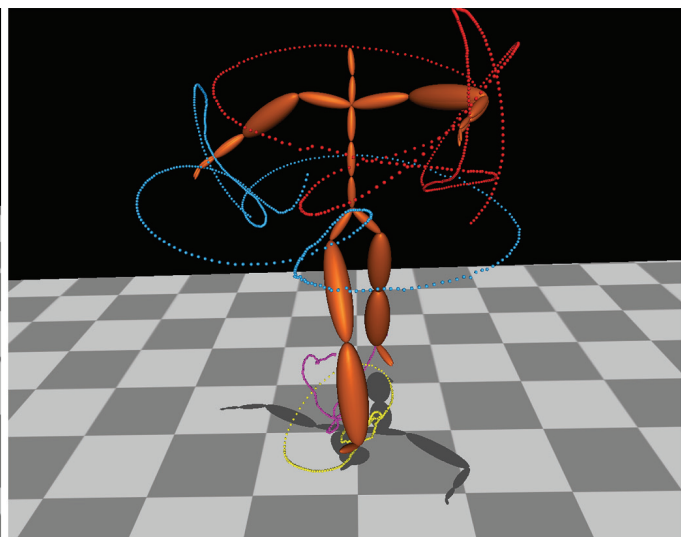
(a) **(b)**

Figure 1. The joint positions of the four handles of two motions without root translations. (a) A walking motion with 316 frames. (b) A dancing motion with 500 frames. Poses in a single motion can span only a narrow area. When users want to obtain poses that are outside that narrow area, the poses might become unnatural.

Finally, the Carnegie Mellon University (CMU) motion-capture database that we use contains data from people with different segment lengths. However, motion data has a negligible impact on training-based IK systems that use only joint rotations to train the model. So, a pose will still look natural even if we scale the character's segments up or down.

Character Posing with NAT-IK

Users can provide a database containing various motion types or use the freely available CMU database (http://mocap.cs.cmu.edu). Our system intends to

efficiently learn a *prior model* from the database. A prior model is a model being trained from a database, so that prior knowledge is maintained in that model. With such a model, new believable information can be inferred. After the system learns the model, users can interactively pose the character by dragging user-defined IK handles. Typically, we set a character's end effectors as IK handles. However, an IK handle can be any joint of the character model. Our system automatically generates natural-looking poses that satisfy user constraints.

Such a system poses two main challenges:

Table 1. Different sparse approximations.

Method	Training time	Training capacity (no. of frames)
Full Gaussian processes (GPs)	36 hrs.	2,000
Deterministic training conditions (DTCs)	9.6 min.	11,000
Partially independent training conditions (PITCs)	19.2 min.	8,000
Fully independent training conditions (FITCs)	23 min.	8,000
Informative vector machine (IVM) with Gaussian process latent variable model (GPLVM)	11.2 hrs.	8,000
IVM with scaled GPLVM (SGPLVM)	18.3 hrs.	8,000

- how to learn a prior model from a database containing millions of frames, and
- how to maintain an interactive posing speed.

We address the first challenge through *representative-frame-set selection*. To meet the second challenge, we use FITC approximation.

To represent the root orientation and joint rotations, our system uses the quaternion logarithm.[5] Quaternions are free of gimbal lock and provide stable interpolation, which are important for calculating the mean and variance during training. A sequence of poses can describe an articulated figure's motion. The root joint's translation and orientation, along with the other joints' orientations, specify a pose. Suppose that the pose at time t is represented by $\mathbf{m}(t) = (\mathbf{p}(t), \mathbf{q}_1(t), ..., \mathbf{q}_n(t))^T$, where $\mathbf{p}(t) \in \mathbf{R}^3$ and $\mathbf{q}_1(t) \in \mathbf{S}^3$ represent the root joint's translation and orientation. $\mathbf{q}_i(t) \in \mathbf{S}^3$ represents the ith joint's orientation for $2 \le i \le n$, where n is the number of joints.

Our IK process has three stages. First, select a representative frame set from a motion database. Second, learn a prior model from that frame set. Finally, solve an optimization problem by combining the prior model and the constraints. We need to perform the first two steps only once.

Selecting a Representative Frame Set

A motion database can contain millions of frames. We can train a full GP model on only 2,000 frames with current desktop PCs. Sparse approximation can improve the training capacity, but our approach can't accommodate more than 10,000 frames, according to our experiments (see Table 1). Fortunately, because the database contains many similar poses, we don't need to include all the poses to train the model. This strategy can dramatically decrease training time. One potential issue is that the strategy breaks down the motions' continuity. However, this breakdown isn't a problem for our system because we focus on editing poses rather than generating smooth motions.

How do you eliminate similar poses and select poses from a large database that are differ-
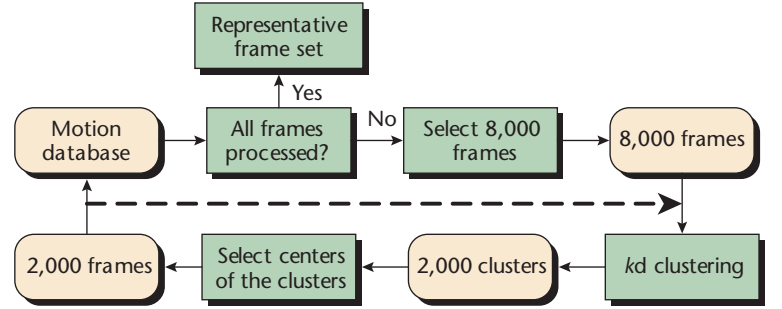
Figure 2. Adaptive representative-frame-set selection. This process removes redundant poses to speed up prior-model training and the synthesis of inverse-kinematic (IK) poses.

ent enough from each other to train the model? A naive solution is to process the database frame-by-frame. For each frame, calculate the pose difference to the processed frames, then discard the current frame if the distance isn't large enough. This solution has $O(N^2)$ complexity, making the calculation impossible with databases containing millions of frames.

To decrease computational complexity, we use a kd-clustering algorithm, which is an efficient implementation of the popular k-means algorithm.[6] It uses a kd-tree to maintain a subset of the candidate centers. Because kd clustering doesn't require an update for each clustering stage, it runs much faster than the k-means algorithm.

Figure 2 illustrates representative-frame selection. Basically, this process randomly selects 8,000 frames from the database and assembles them into 2,000 clusters. Then, it selects the centers of the 2,000 clusters as the current representative frame set and fetches another set of unprocessed 8,000 frames from the database. The process combines the selected 2,000 frames with the new 8,000 frames, then selects a new representative frame set from the 10,000 frames. This process repeats until it has processed all the frames in the database. We call this algorithm *adaptive representative-frame-set selection*.

The algorithm selects the number of representative frames according to the reconstruction-error and IK-speed curves that we obtained in an experiment (see Figure 3). In this experiment, we

Gaussian Processes for Large Datasets

Gaussian processes (GPs) are popular and powerful for nonlinear nonparametric regression and classification. Unfortunately, they don't scale well with large datasets. Researchers have proposed several methods to accelerate GPs. These methods fall into two categories: sparse approximation and matrix-vector multiplication (MVM).[1]

Sparse-approximation methods use the subsets of data or the inducing variables to approximate the training variables' distribution.[2,3] One such method is the *informative vector machine* (IVM), proposed by Neil Lawrence[4] and adopted by style-based IK (see the main article). One potential problem with IVM is that it depends only on the active dataset, which changes as the optimization proceeds. Determining when convergence has occurred can be difficult. IVM provides a rough approximation of the training conditions.

MVM methods combine linear-equation solvers with fast Gaussian transformation, to calculate the most expensive matrix inverse. MVM is efficient when the dimensions of both the input and output data are low. Unfortunately, motion data has high-dimensional spaces. Although the input feature space's dimensions can be reduced by principle component analysis, a Gaussian process latent variable model, or other techniques, the output space's dimension still will be high. This condition implies that MVM isn't suitable for IK problems. To cope with large motion databases, we finally selected *fully independent training conditional approximations*.[1,4]

References

1. J.Q. Candela, C.E. Rasmussen, and C.K.I. Williams, "Approximation Methods for Gaussian Process Regression," *Large-Scale Kernel Machines*, L. Bottou et al., eds., MIT Press, 2007, pp. 203–223.
2. M. Gleicher, "Motion Editing with Spacetime Constraints," *Proc. 1997 Symp. Interactive 3D Graphics* (I3D 97), ACM Press, 1997, pp. 139–148, 195.
3. J.Q. Candela and C.E. Rasmussen, "A Unifying View of Sparse Approximate Gaussian Process Regression," *J. Machine Learning Research,* vol. 6, 2005, pp. 1939–1959.
4. N.D. Lawrence, "Learning for Larger Datasets with the Gaussian Process Latent Variable Model," *Proc. 11th Int'l Conf. Artificial Intelligence and Statistics* (AISTATS 07), 2007; www.stat.umn.edu/~aistat/proceedings/data/papers/031.pdf.
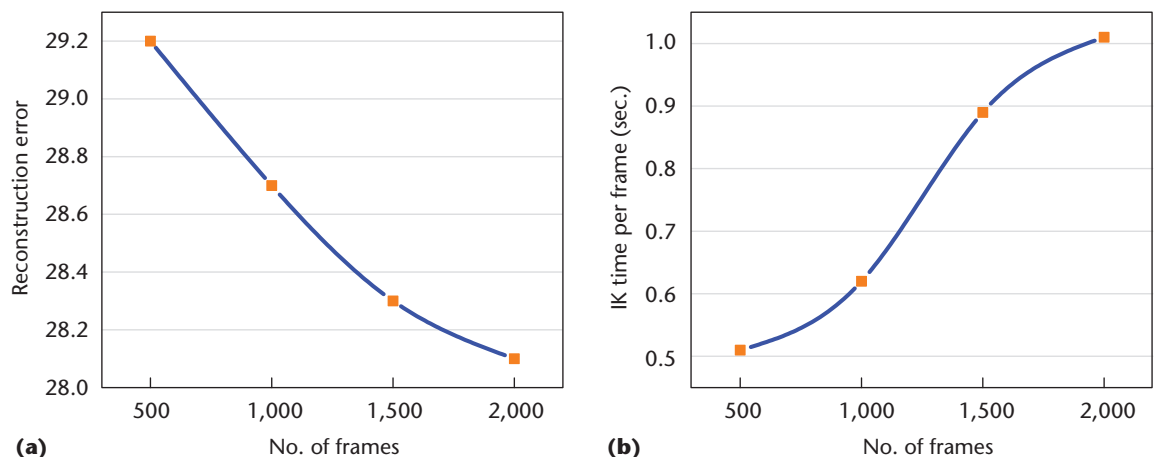
Figure 3. The (a) reconstruction error and (b) IK processing time according to the number of frames in the representative frame set. On the basis of these two curves, we chose 2,000 frames as the representative frame set.

generated different numbers of frames for the representative frame set. Then, we synthesized 100 poses by setting the IK handles' positions to the same positions as those of the ground-truth poses. Next, we calculated the average Euclidean distance of joints. We used six latent dimensions, 25 iterations, and 200 active points for the FITC approximation. We randomly selected 100 poses and used the positions of these poses' end effectors as constraints. We then ran NAT-IK to generate 100 poses based on these constraints. We measured the reconstruction error and IK time by comparing the NAT-IK-generated poses and the randomly selected poses. On the basis of the results, we decided to use 2,000 representative frames to maintain interactive speed and tolerable error.

Learning a Model

We use GPs to train our model because they're widely accepted and have proven efficient in many applications. (For more on GPs for large datasets, see the related sidebar.)

Using approximation methods to accelerate GPs to work with large datasets is an important issue

Table 2. Reconstruction errors for different sparse approximations.

Full GP	DTC	PITC	FITC	IVM (GPLVM)	IVM (SGPLVM)
0.10	0.45	0.38	0.30	0.45	0.40

that many researchers have addressed. The time complexity of training a full GP model is $O(N^3)$. You can reduce the training time to $O(MN^2)$ by using inducing variables. (Here, N is the training dataset's size and M is the number of inducing variables.) We examined three approximation methods: *deterministic training conditions* (DTC), *partially independent training conditions* (PITC), and FITC.[3,4] Table 1 compares them. We measured only these three approximation methods because they're the most accurate and because we require a fast, accurate approximation.

We could have also used linear-equation solvers to do the most expensive matrix inversion during training. In addition, we could have further accelerated this process by using the *improved fast Gaussian transform*.[3] However, this approach wouldn't have helped solve the IK problem because the problem's output space would still be high.

We finally chose FITC approximation on the basis of its accuracy, training speed, and prediction speed. Table 2 shows the reconstruction error for the sparse methods, comparing DTC, PITC, and the *informative vector machine* (IVM) with the *Gaussian process latent variable model* (GPLVM) and the *sparse GPLVM* (SGPLVM). We selected a walking motion containing 500 frames. We used six latent dimensions and the 100 active frames. The results show that, of the tested methods, FITC produces the most accurate model, followed by PITC. Both FITC and PITC outperform IVM.

GPs and FITC approximation. Given a training data set $D = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, n\}$ of n pairs of inputs $\mathbf{x}_i$ and noisy outputs $\mathbf{y}_i$ with $\mathbf{y}_i = f(\mathbf{x}_i) + \varepsilon_i$, where $\varepsilon \sim N(0, \sigma_{noise}^2)$, GP regression aims to find the predictive distribution of the function values f_* at some new locations $\mathbf{x}_*$. ε is the Gaussian noise. σ^2 is the variance of the Gaussian distribution $N(0, \sigma^2)$. The prediction mean and variance are

$$\bar{f}_* = \mathbf{k}_*^T \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y}$$

$$V\left[f_*\right] = k\left(\mathbf{x}_*, \mathbf{x}_*\right) - \mathbf{k}_*^T \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{k}_*,$$

where $\mathbf{K}$ is a kernel matrix with $\mathbf{K}[i][j] = k(\mathbf{x}_i, \mathbf{y}_i)$ and $\mathbf{k}_* = k(\mathbf{x}_*, \mathbf{x})$. $\mathbf{I}$ is an identity matrix. We use the kernel matrix

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \alpha \exp\left(-\frac{\gamma}{2} \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) + \beta \delta_{ij} + \nu,$$

where α is the variance and γ is the inverse width of the RBF kernel, β is a noise term, ν is the bias term, and δ_{ij} is a Dirac delta function.

We express FITC's prediction distribution as

$$q\left(\mathbf{f}_* \mid \mathbf{y}\right) = N\begin{pmatrix} \mathbf{Q}_{*,\mathbf{f}}\left(\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \Lambda\right)^{-1} \mathbf{y}, \mathbf{K}_{*,*}, \\ -\mathbf{Q}_{*,\mathbf{f}}\left(\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \Lambda\right)^{-1} \mathbf{Q}_{\mathbf{f},*} \end{pmatrix},$$

where $\mathbf{Q}_{\mathbf{a},\mathbf{b}} = \mathbf{K}_{\mathbf{a},\mathbf{u}} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{K}_{\mathbf{u},\mathbf{b}}$. $\mathbf{K}_{\mathbf{p},\mathbf{q}}$ is a kernel matrix between two inputs $\mathbf{p}$ and $\mathbf{q}$. $\mathbf{u}$ is the inducing variable.[3] $\Lambda = diag\left[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{Q}_{\mathbf{f},\mathbf{f}} + \beta I\right]$.

The log likelihood objective function we use to learn the model with FITC approximation is[7]

$$\log p\left(\mathbf{Y} \mid \mathbf{X}, \mathbf{X}_\mathbf{u}, \theta\right) = -\frac{d}{2}\log\left(2\pi\right) - \frac{d}{2}\log\left|\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \mathbf{H}\right| \\ -\frac{1}{2} tr\left(\mathbf{Y}\mathbf{Y}^T\left(\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \mathbf{H}\right)^{-1}\right),$$

where $\mathbf{X}$ is the original training data's latent variable and $\mathbf{X}_\mathbf{u}$ is the inducing variable's latent variable. $\mathbf{Y}$ is an N-by-D matrix. N is the number of frames, and D is the data dimension given by $D = 3J$, where J is the number of joints. $\mathbf{H}$ is a diagonal matrix defined as $\mathbf{H} = diag\left(\beta^{-1} + \mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{u}} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$.

Synthesizing poses with constraints. We've formulated pose synthesis as a constrained-optimization problem:

$$\arg \min_{\mathbf{x},\mathbf{y}} L_{IK}\left(\mathbf{x}, \mathbf{y}\right)$$

$$\text{such that } C\left(\mathbf{f}\right) = 0,$$

where

$$L_{IK}\left(\mathbf{x}, \mathbf{y}\right) = \frac{\|\mathbf{y} - \mathbf{f}\left(\mathbf{x}\right)\|^2}{2\sigma^2\left(\mathbf{x}\right)} + \frac{D}{2}\ln \sigma^2\left(\mathbf{x}\right) + \frac{1}{2}\|\mathbf{y} - \mathbf{y}_p\|^2 \qquad (1)$$

$$\mathbf{f}\left(\mathbf{x}\right) = \mu + \mathbf{q}_{*,\mathbf{f}}\left(\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \Lambda\right)^{-1} \mathbf{Y}$$

$$\sigma^2\left(\mathbf{x}\right) = k\left(\mathbf{x}, \mathbf{x}\right) - \mathbf{q}_{*,\mathbf{f}}\left(\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \Lambda\right)^{-1} \mathbf{q}_{*,\mathbf{f}}^T.$$

L is the likelihood function as well as the objective function, and C refers to the constraints. D is the dimension of the original training dataset, μ is the training data's mean, $\mathbf{y}_p$ is the $\mathbf{y}$ of the previous pose that users have edited, and $\mathbf{q}_{*,f} = \mathbf{K}_{*,\mathbf{u}} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{K}_{\mathbf{u},*}$. (For the derivatives for Equation 1, see the "Derivatives of the IK Objective Function" sidebar.)

We add a smoothness term, $1/2\|\mathbf{y} - \mathbf{y}_p\|^2$, at the objective function's end. This term limits the changes in the neighboring poses and smoothes the

Derivatives of the IK Objective Function

Here are the derivatives for Equation 1 in the main article:

$$\frac{\partial L_{IK}}{\partial \mathbf{x}} = -\left(\mathbf{y} - \mathbf{f}(\mathbf{x})\right)\frac{\partial \mathbf{f}(\mathbf{x})^T}{\partial \mathbf{x}}\Bigg/ \sigma^2(\mathbf{x})$$

$$+ \frac{\partial \sigma^2(\mathbf{x})}{\partial \mathbf{x}}\left[D - \frac{\|\mathbf{y} - \mathbf{f}(\mathbf{x})\|^2}{\sigma^2(\mathbf{x})}\right]\Bigg/ 2\sigma^2(\mathbf{x})$$

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial \mathbf{q}_{*,\mathbf{f}}}{\partial \mathbf{x}}\left(\mathbf{Q}_{\mathbf{f},\mathbf{f}} - \Lambda\right)^{-1}\mathbf{Y}$$

$$\frac{\partial \mathbf{q}_{*,\mathbf{f}}}{\partial \mathbf{x}} = \frac{\partial \mathbf{K}_{*,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}}{\partial \mathbf{x}} = \frac{\partial \mathbf{K}_{*,\mathbf{u}}}{\partial \mathbf{x}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}$$

$$\frac{\partial \sigma^2(\mathbf{x})}{\partial \mathbf{x}} = -2\mathbf{q}_{*,\mathbf{f}}\left(\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \Lambda\right)^{-1}\frac{\partial \mathbf{q}_{*,\mathbf{f}}^T}{\partial \mathbf{x}}$$

$$\frac{\partial L_{IK}}{\partial \mathbf{y}} = \frac{\left(\mathbf{y} - \mathbf{f}(\mathbf{x})\right)}{\sigma^2(\mathbf{x})} + \mathbf{y} - \mathbf{y}_P$$

For an explanation of the symbols, see the sections "GPs and FITC approximation" and "Synthesizing poses with constraints" in the main article.

Figure 4. The trajectories of the selected representative frames from the database. The left wrist is red, the right wrist is light blue, the left foot is purple, and the right foot is yellow. NAT-IK's trajectories span a much larger space than Style-IK, making NAT-IK more robust.
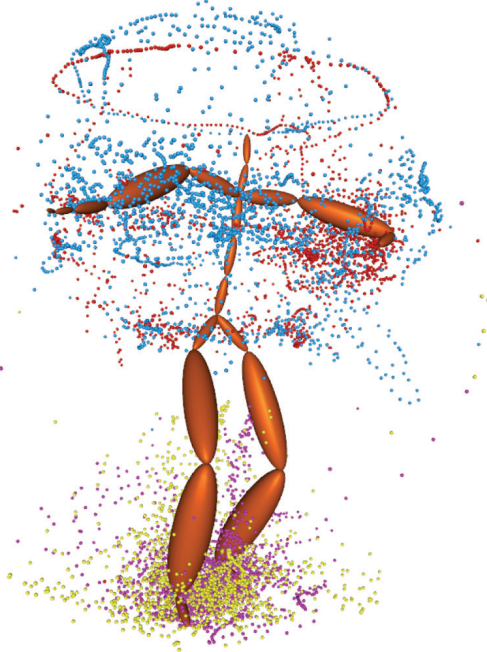
neighboring poses. The objective function's first term evaluates the predicted mean's difference from pose **y**. The second term evaluates the prediction error.

The constraints are the selected joints' positions or orientations, as the user specified.

Results

We evaluated NAT-IK on a Dell Precision 390 workstation with a dual-core 2.13-GHz CPU and 3.5 Gbytes of RAM. We implemented the model learning with Matlab code and coded the pose synthesis with C++. We used Blitz++ for matrix storage and the Intel Math Kernel Library for matrix operations. For a video demonstration, see http://sites.google.com/site/wuxiaomao/natik or http://doi.ieeecomputersociety.org/10.1109/MCG.2009.111.

We trained the model with the scaled conjugate gradient (SCG) algorithm[8] and solved pose synthesis with the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method[9] by converting the constrained optimization into an unconstrained optimization. Our original objective function's weight was 1.0; the constraint part's weight was 10.0. We set the root movement's weight at 200 for two reasons. First, we prefer not to move the root joint unless doing so is necessary to reach the target positions. Second, the root translation vector's derivatives are much smaller than the quaternion logarithm.

Selecting Training Data

From the CMU database, we randomly selected 1,300 motions containing 2,510,491 frames. We selected 2,000 of these motions, using adaptive representative-frame-set selection.

Figure 4 visualizes the selected frames' trajectories. Obtaining such a widely spanned space is impossible without processing a large motion database.

Data selection required approximately 20 hours. Fortunately, as we mentioned before, we need to run this process only once. Then, we can use it indefinitely.

Model Learning

We trained the prior model with the 2,000 representative frames. We then saved the model parameter θ, the latent variables **X**, and the latent inducing variable $\mathbf{X_u}$.

With the Matlab code, learning the model required 8 minutes and 12 seconds. To train the same dataset required 24 hours for SGPLVM. For NAT-IK and SGPLVM, we used 100 iterations, 12 latent dimensions, and 200 inducing variables. Figure 5 shows the reconstruction errors and IK processing time for NAT-IK according to the number of latent dimensions; Figure 6 illustrates the trained latent space. Unlike with style-based IK, the latent space for NAT-IK isn't smooth but spans a wide space, as we expected. This rough spacing is because our approach fetches the active training frames from different types of motions.

Interactive Character Posing

The average time for synthesizing a pose was 1 second. Style-based IK took 8 seconds for 2,000
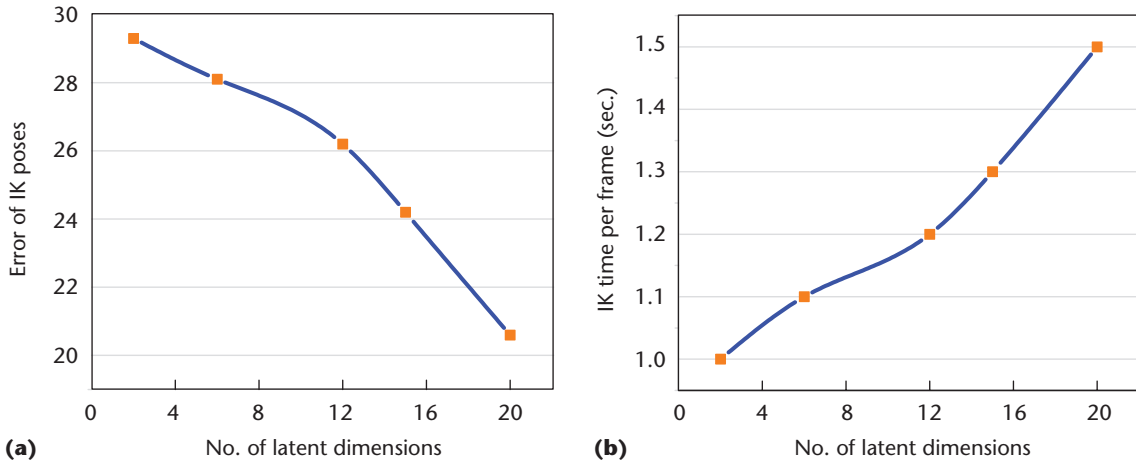
Figure 5. The (a) reconstruction errors and (b) IK processing time for NAT-IK according to the number of latent dimensions. We chose the latent dimension to be 12, which is a trade-off between the error and speed.

training frames and approximately 1 second for 600 frames.

We demonstrated only five handles with position constraints; however, we could easily add more handles. We tested the performance with different numbers of handles. The results show that the number of handles has negligible influence on the IK pose-synthesizing speed. We obtained the handle positions from captured motion data.

Table 3 shows the reconstruction error and pose-synthesizing time for three IK algorithms. We randomly selected 100 poses from the database, set the five handles' positions according to the selected poses, and reconstructed these poses. We calculated the error by adding the Euclidean distance between all the joints over the 100 frames, then dividing by the sum of the number of frames and joints. In this experiment, we trained SGPLVM with a boxing motion containing 500 frames, using representative frames from the database. For all three algorithms, we used the same values for the number of latent dimensions (12), active frames (200), and SGPLVM iterations (25). Both NAT-IK and style-based IK were initialized to a T-pose at the beginning of optimization.

Figure 7 illustrates the results for style-based IK and NAT-IK. Style-based IK fails to generate natural poses because the original pose differs from the poses we used to train the SGPLVM model. However, if the pose is similar to the training poses, style-based IK functions effectively (see Figure 8).

We also compared NAT-IK with traditional IK (see Figure 9), which we also implemented with the L-BFGS code. We didn't use any energy terms or joint limits. Using joint limits changes the posing speed from 70 milliseconds to 4 seconds but still produces unnatural poses. Energy terms such as minimal joint angles might help, but they don't use statistical information from real human poses.
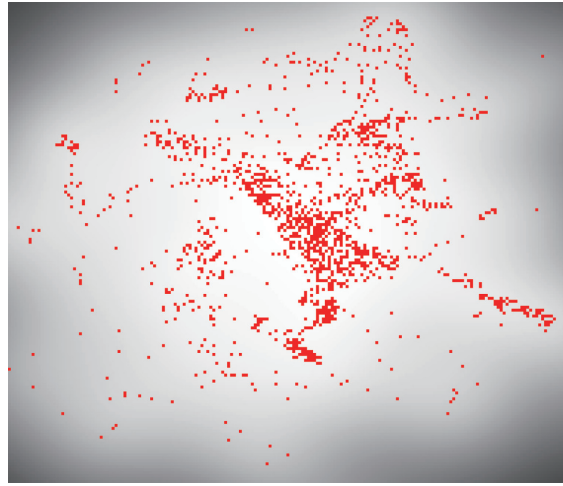
Figure 6. The latent space that NAT-IK generated. Training samples span most of the latent space, making NAT-IK more robust, and flexible to new circumstances.

Table 3. Results for three inverse-kinematics (IK) algorithms.

	Traditional IK	Style-based IK	NAT-IK
Reconstruction error (mm)	44.8	34.3	29.2
Pose-synthesizing time (sec.)	0.07	2.40	0.50

Discussion

Here, we look at several major concerns that readers might have and discuss NAT-IK's limitations.

Why Does NAT-IK Work?

An intuitive explanation of why NAT-IK produces different results from style-based IK is that by discarding continuity and smoothness, and training a set of representative frames selected from a large motion database, we can make the confident area of the latent space much larger than style-based IK can. Because we abandon style and continuity, we can select diverse poses from a large database and
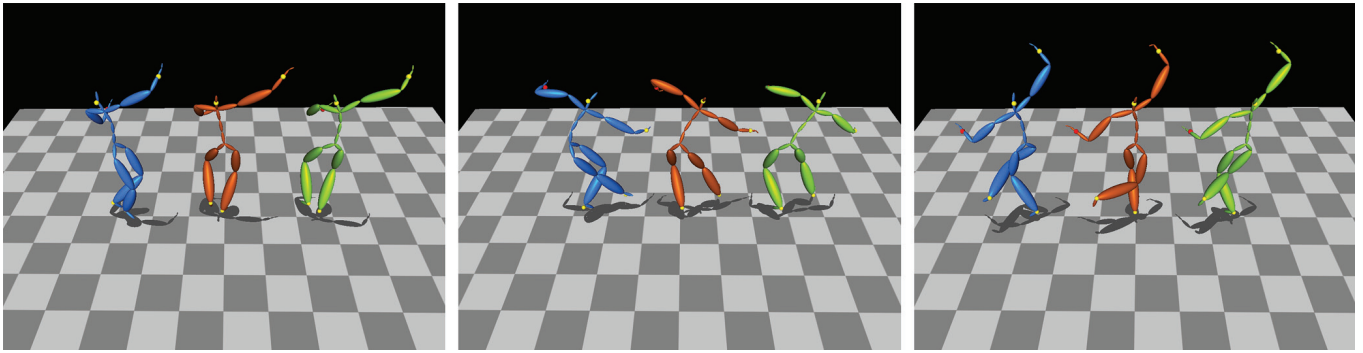
Figure 7. NAT-IK compared with style-based IK for varied, randomly selected poses. Style-based IK generates unnatural poses when the IK handles are far from those of the training motion. We generated these poses by setting the handle positions to those of the original motion. The blue, red, and green figures represent the results for style-based IK, the ground truth, and our method.
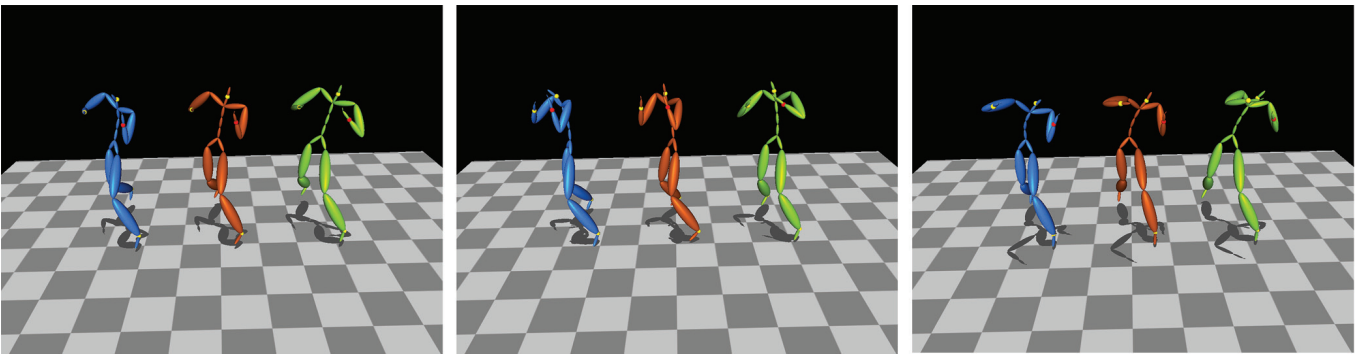
Figure 8. NAT-IK compared with style-based IK for the motion used to train the sparse Gaussian process latent variable model (SGPLVM). Style-based IK has a smaller reconstruction error than that of NAT-IK because these poses are in the confident area of SGPLVM's latent space. The blue, red, and green figures represent the results for style-based IK, the ground truth, and our method.
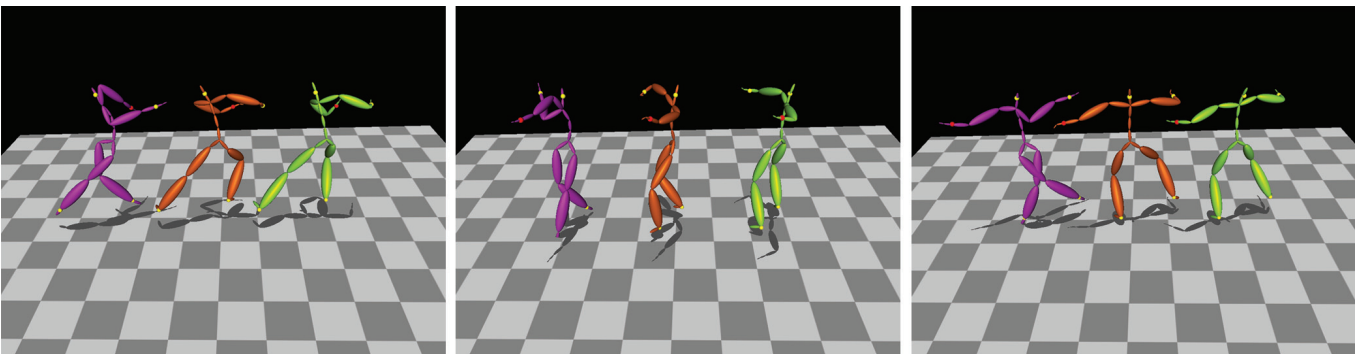
Figure 9. NAT-IK compared with traditional IK for varied, randomly selected poses. Traditional IK generates unnatural poses because it has no prior information. The purple, orange, and green figures represent the results of traditional IK, the ground truth, and our method.

learn a better model from it. This process is why NAT-IK produces poses that are different from and generally more robust than those that style-based IK produces.

NAT-IK versus Style-Based IK

Style-based IK is more general than NAT-IK and is more suitable for generating smooth motions. NAT-IK is mainly for per-frame IK. Style-based IK can't deal with large datasets; it can generate natural poses only if the target poses are similar to the training poses. As we mentioned before, NAT-IK can deal with large datasets and generate natural poses in a much wider, human-reachable space. Because style-based IK requires smooth training poses, it can't handle sparse poses.

Training Discrete Poses

The latent space becomes nonsmooth if we train from discrete poses instead of continuous poses. However, because the latent space is dense enough, the optimization still can find a good path to reach the right position. We've checked the reconstruction errors and found that the optimization works

effectively. We haven't encountered any convergence or overfitting problems for FITC approximation.

Soft Constraints of the Optimization Process

Converting hard constraints into soft constraints worked reasonably well in our experiments. We don't use hard constraints because they dramatically slow down the IK speed. Using soft constraints greatly improves the performance and converges well.

Generalization of NAT-IK

How can we use the learned model for arbitrary topology of human-like skeletons? One solution is to define key-joint correspondence manually and to map other nonkey joints by interpolation. In this situation, we can use any motion-retargeting methods.

Limitations

In our approach, the pose-synthesizing speed is interactive (1 sec.) but not real-time. However, we didn't optimize our code, so this speed has much potential to increase. One way to accelerate NAT-IK might be to employ precomputation and fast matrix multiplication.

IK algorithms are fundamental in computer animation. However, designing energy functions that can generate natural poses for traditional IK algorithms is difficult. Style-based IK solves this problem by learning a prior model from motions. However, it might fail to generate natural poses when the desired poses differ considerably from the limited training poses. As we've shown, NAT-IK overcomes these limitations. It can relieve animators from time-consuming, back-and-forth, IK-pose adjustment. So, it's useful in automated applications such as games and virtual worlds.

References

1. M. Gleicher, "Motion Editing with Spacetime Constraints," *Proc. 1997 Symp. Interactive 3D Graphics* (I3D 97), ACM Press, 1997, pp. 139–148, 195.
2. K. Grochow et al., "Style-Based Inverse Kinematics," *ACM Trans. Graphics*, vol. 23, no. 3, 2004, pp. 522–531.
3. J.Q. Candela, C.E. Rasmussen, and C.K.I. Williams, "Approximation Methods for Gaussian Process Regression," *Large-Scale Kernel Machines*, L. Bottou et al., eds., MIT Press, 2007, pp. 203–223.
4. J.Q. Candela and C.E. Rasmussen, "A Unifying View of Sparse Approximate Gaussian Process Regression," *J. Machine Learning Research*, vol. 6, 2005, pp. 1939–1959.
5. F.S. Grassia, "Practical Parameterization of Rotations Using the Exponential Map," *J. Graphics Tools*, vol. 3, no. 3, 1998, pp. 29–48.
6. T. Kanungo et al., "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, 2002, pp. 881–892.
7. N.D. Lawrence, "Learning for Larger Datasets with the Gaussian Process Latent Variable Model," *Proc. 11th Int'l Conf. Artificial Intelligence and Statistics* (AISTATS 07), 2007; www.stat.umn.edu/~aistat/proceedings/data/papers/031.pdf.
8. M.F. Møller, "A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning," *Neural Networks*, vol. 6, no. 4, 1993, pp. 525–533.
9. D.C. Liu and J. Nocedal, "On the Limited Memory BFGS Method for Large Scale Optimization," *Mathematical Programming*, vol. 45, no. 3, 1989, pp. 503–528.

Xiaomao Wu *is a senior application engineer in the R&D department of Crytek's Frankfurt studio. His research interests include computer animation, VR, and game engine design. Wu has a PhD in computer science from Shanghai Jiao Tong University. He's on the* ACM Computers in Entertainment *editorial board. Contact him at wu.xiaomao@gmail.com.*

Maxime Tournier *is a PhD student at the Université de Grenoble, INRIA Rhône-Alpes. His research interests include motion analysis, motion synthesis, and physically based animation. Tournier has an MD in computer sciences from the Grenoble Institute of Technology. Contact him at maxime.tournier@inrialpes.fr.*

Lionel Reveret *is a senior research scientist at INRIA. His research interests include video-based motion capture, motion compression, and animation of animals and plants. Reveret has a PhD in computer sciences from the Grenoble Institute of Technology. Contact him at lionel.reveret@inria.fr.*

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.

$$\therefore$$

5.7 Locomotion Skills for Simulated Quadrupeds

Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, Michiel van de Panne ACM Transactions on Graphics, Proceedings of SIGGRAPH'11, Vancouver, Canada, aug 2011.

To appear in the ACM SIGGRAPH 2011 conference proceedings

Locomotion Skills for Simulated Quadrupeds

Stelian Coros[1,2]   Andrej Karpathy[1]   Ben Jones[1]   Lionel Reveret[3]   Michiel van de Panne[1] *

[1]University of British Columbia   [2]Disney Research Zurich   [3]INRIA, Grenoble University, CNRS
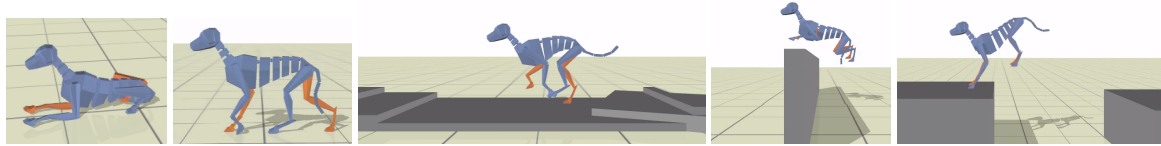
Figure 1: *Real-time physics-based quadruped simulations of gaits (walk, trot, canter, transverse gallop, pace, rotary gallop), gait transitions, sitting and standing up, targeted jumps, and jumps on-to and off-of platforms.*

Abstract

We develop an integrated set of gaits and skills for a physics-based simulation of a quadruped. The motion repertoire for our simulated dog includes walk, trot, pace, canter, transverse gallop, rotary gallop, leaps capable of jumping on-and-off platforms and over obstacles, sitting, lying down, standing up, and getting up from a fall. The controllers use a representation based on gait graphs, a dual leg frame model, a flexible spine model, and the extensive use of internal virtual forces applied via the Jacobian transpose. Optimizations are applied to these control abstractions in order to achieve robust gaits and leaps with desired motion styles. The resulting gaits are evaluated for robustness with respect to push disturbances and the traversal of variable terrain. The simulated motions are also compared to motion data captured from a filmed dog.

1 Introduction

Quadrupedal animals form an important part of the world around us. It is therefore not surprising that cats, dogs, mice, horses, donkeys, elephants, and other animals, real or mythical, make regular appearances in games, films, and virtual world simulations. Games which use interactive quadruped animation include *Zoo Tycoon*, *Red Dead Redemption*, *Cabela's African Safari*, and *Assassin's Creed*. Example films include *Lord of the Rings*, *Chronicles of Narnia*, and *Cats and Dogs*, to name but a few. Quadruped movement is extremely rich because of the many possible gaits and the variations in body size and body proportions, e.g., from shrews to elephants. There exist a multitude of ways in which the skeleton and legs can support the locomotion. The difficulty of modeling such a diverse set of motions is further compounded by the paucity of available motion capture data.

As was first proposed two decades ago [Raibert and Hodgins 1991], the use of forward dynamics simulation with suitable controllers offers one possible approach for creating interactive, reactive quadruped motions. We build on this general approach with the following contributions:

---

* scoros|andrejk|jonesben|van@cs.ubc.ca, lionel.reveret@inria.fr

- We develop several abstractions for use in quadruped simulation, including a dual leg frame model, a flexible abstracted spine, and the extensive use of internal virtual forces. These form part of a flexible vocabulary for the design of quadruped motions.

- We demonstrate the creation of walk, trot, pace, canter, and transverse and rotary gallop gaits of varying speeds for a simulated dog using these control abstractions. The gaits are automatically tuned (optimized) to satisfy a variety of objectives. We compare our motions with captured motion for a dog. We evaluate the robustness of the gaits with respect to gait transitions, pushes, and unexpected steps.

- We develop a flexibly parameterized jump that can be executed from various initial trotting speeds. This allows the simulated quadruped to jump onto and off of platforms, jump over obstacles, and jump over gaps. We further develop controllers for sitting, lying-down, and standing up.

2 Related Work

A mix of kinematic and dynamic methods have been applied to quadruped animation, dating back over a quarter century. A comprehensive recent survey of quadruped animation work is given in [Skrba et al. 2008]. The following survey is heavily focused on controller-based methods, and even then it is selective because of the breadth of previous work in this area.

Procedural and trajectory-based methods: The early work of Girard and Maciejewski [1985] proposes the use of gait patterns, foot location splines, inverse kinematics, and body location that is constrained by simplified body dynamics. Blumberg and Galyean [1995] develop a multi-layer kinematic approach as the simulated motor system of a dog, with a focus on supporting higher level behaviors. The game of Spore [Hecker et al. 2008] develops methods for generating procedural animation for arbitrary legged creatures, including locomotion patterns. Torkos and van de Panne [1998] apply trajectory optimization techniques to an abstracted quadruped model to obtain motions that are compatible with given foot locations and timing patterns. Wampler and Popović [2009] develop a two-level optimization procedure for physics-based trajectories of periodic legged locomotion and use it to explore connections between form and function. Kry et al. [2009] explore the use of modal deformations as the basis for developing periodic gait patterns directly from the geometry of a dog model.

2D forward dynamic simulations: The dynamic simulation of quadruped gaits is a shared goal across animation, robotics, and

biomechanics. Physics-based simulations are typically better suited for modeling unscripted interactions with the environment than kinematic models. 2D sagittal plane simulations can reveal much about the nature of quadruped gaits without dealing with the full intricacies of 3D motion, and have thus often been used for the development and analysis of quadruped gaits and control strategies. Van den Bogert [1989] develops a 2D rigid body model with significant aspects of the motion constrained so as to follow given motion and ground-reaction force data. Marhefka et al. [2003] develop a control strategy based on fuzzy logic to produce simulated planar gallops. Krasny and Orin [2004] use a search algorithm to explore the space of open-loop 2D gallops. Herr and McMahon [2001] develop and analyze feedforward and feedback strategies for the transverse gallop of a horse, as tested using a 10-link planar simulation. Wong and Orin [1995] develop control strategies for quadruped standing jumps using a simplified planar model.

3D forward dynamic simulations of quadruped gaits are introduced by Raibert and Hodgins [1991], who develop control strategies for trotting, bounding, and galloping gaits for a robot quadruped with a rigid body and extensible legs. Kokkevis et al. [1995] present a hybrid kinematic-and-dynamic controller and simulation for a 3D dog with a rigid trunk that can walk and trot using a linear programming framework. Ringrose [1996] demonstrates in simulation that gaits such as transverse and rotary gallops can be self-stabilizing for appropriate body and leg design and circular foot profiles. Van de Panne [1996] explores the use of optimized open-loop, passively-stable control strategies to generate dynamically-simulated 3D locomotion for a wide-stanced cat, including walk, trot, bound, and rack gaits. Krasny and Orin [2006] employ a multi-objective optimization to develop stable gallops for a 9-link 3D simulation. Our work improves upon the state of the art in 3D quadruped simulation in a number of ways, as outlined at the end of this section.

Quadruped Robots: Numerous quadruped robots have been developed, along with control strategies that are compatible with their specific mechanical design. The Raibert quadruped [Raibert 1986] and BigDog [Buehler et al. 2005; Playter et al. 2006] provide seminal demonstrations of movement that is highly robust to pushes and many different types of terrain, e.g., snow, mud, and steep hills. The most developed gait simultaneously moves diagonally-opposite leg pairs, as in a trotting gait. The SCAMPER robot provides an early demonstration of a dynamic and symmetric 'bounce' gait [Furusho et al. 2002]. The SCOUT robot is a small robot capable of walking, climbing, and galloping [Poulakakis et al. 2005] using one-degree-of-freedom legs. Walking gaits have been automatically optimized for the Sony AIBO robot using policy gradient methods [Kohl and Stone 2004]. Numerous control strategies have been developed for the LittleDog robot, e.g., [Kolter et al. 2008; Zucker et al. 2010]. These have typically focused on slower gaits and rough terrain that demands careful motion planning. CPG-based control strategies are demonstrated in [Tsujita et al. 2001] and are developed for dynamic walking over uneven terrain for the Tekken2 robot [Kimura et al. 2007].

Biped Control: The development of physics-based animated characters that can walk and run has been a difficult problem that has recently seen significant progress, e.g., [Hodgins et al. 1995; Laszlo et al. 1996; Yin et al. 2007; Sok et al. 2007; Muico et al. 2009; Ye and Liu 2010; Lee et al. 2010; Coros et al. 2010; Wu and Popović 2010; de Lasa et al. 2010]. Faloutsos et al. [2001] develop a framework for the serial composition of controllers for a simulated human model. A comprehensive review of the extensive prior art on the problem of biped control is beyond the scope of this paper. The ideas developed for biped control provide insights and inspiration for quadruped control. However there are many open questions in extending these methods to quadrupeds. Which approaches will
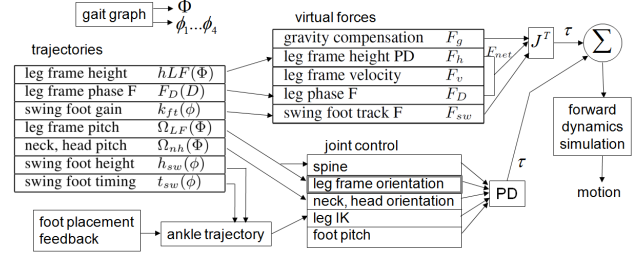
Figure 2: *Controller Overview.*
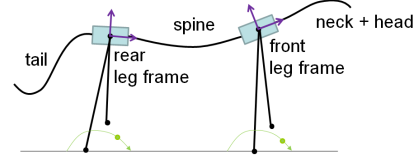
Figure 3: *Abstract quadruped model. The model has separate front and back leg frames which are each use independent foot placement.*

scale well? What are good quadruped motion objectives? How can quadruped motions be authored in the face of a lack of motion capture data? Which abstract models are suitable to use in control computations? How should the flexible spine be modeled? We provide answers to some of these questions in this paper.

Our work: The methods and results developed in this paper can be distinguished from previous work in a number of ways. We develop a flexible abstract spine model which helps achieve more natural motion during fast gaits such as the gallop. The flexible spine connects front and back leg frames, which together form a dual leg frame model, with each leg frame making independent decisions with regard to footstep placement, height control, and pitch control. The controller also exploits gait-specific feed-forward (phase-based) internal virtual forces which are tuned using optimization. The capabilities of a simulated quadruped can be evaluated along any number of dimensions, including the number of supported gaits and gait transitions, their speed, ability to turn, robustness to external perturbations and terrain, similarity to known animal quadruped gaits, and the number of other skills that can be performed, e.g., leaping, sitting, and getting up after a fall. We evaluate our simulated quadruped along many of these dimensions.

3 Quadruped Gait Controllers

A good control representation should be compact, expressive, and provide robust motion when perturbed. Figure 2 provides a structural overview of the various components of the quadruped controller developed in this paper, each of which will be described in more detail shortly. The controller also makes use of the quadruped abstraction shown in Figure 3. As illustrated, this model views the quadruped in terms of front and rear *leg frames* connected by a flexible spine. Various components of the controller are designed directly with this quadruped abstraction in mind, such as the virtual forces, the gait graphs, and various trajectories. Many of the trajectories and parameters that help define the controller are tuned in a separate offline optimization stage which will is detailed in the subsequent section (§4).

The **overall control loop** provides torques to a forward dynamics simulator at every time step, as shown in Figure 2. The forward dy-

namics simulation is treated as a black box; the controller does not exploit any specific knowledge about the equations of motion at any time step. The computed torques arise from two sources: via virtual internal forces that are applied via Jacobian transpose control, and via joint control that compute torques using proportional-derivative (PD) controllers. Torques from these two sources are then summed together. Pseudocode of the overall control loop is provided in the Appendix.

3.1 Gait Controller Details

Understanding the controller requires understanding the types of components from which it is constructed (Figure 2) as well as how these components are used to drive the motion of the abstract quadruped model (Figure 3).

Gait graphs: Quadruped gaits are characterized in large part by the timing and relative phasing of the swing and stance phases of each of the legs. This is captured by the *gait graph* and the overall stride period, T. Figure 4 shows the gait graphs for the six gaits modeled in this paper. Progress made within a stride is defined by the *gait phase*, $\Phi \in [0, 1)$, computed as $\Phi = t/T$, where T is the desired gait period. We further define progress within the stance phase or swing phase for a leg using $\phi \in [0, 1)$, Some aspects of motion are modeled as a function of gait phase, such as the desired height and pitch of the hips. Other aspects are modeled as a function of stance or swing phase, such as the swing foot height trajectories. With the exception of the canter, we obtain the gait graphs from tracking of experimental video data recorded for a dog [Abourachid et al. 2007]. Extracts of the video are included in the video accompanying this paper and experimental gait graphs measured over several cycles of motion are included in the supplemental material. For the pace, we use the timing for the trot, but applied to left and right pairs instead of diagonal pairs. For the canter, we estimate a gait graph based on data provided by Alexander [1984].

Virtual forces are one of the two primary sources of computed torques, as shown in Figure 2. These allow the skeletal structure to be largely abstracted away by specifying coordinated sets of internal torques. A virtual force, F, is applied by using the Jacobian transpose to compute the required internal torques according to $\tau = J^T F$ [Paul 1981]. The virtual forces can be applied at specific points or can be applied to derived variables such as the center of mass of a collection of links [Sunada et al. 1994; Pratt et al. 2001]. A *base link* must be specified for each virtual force. Figure 5 illustrates the terminology we shall use to describe the application of virtual forces. In the shown example, the virtual force specifies torques in the back that help to raise the front of the body.

Joint control provides the second source of computed torques. With the exception of the hip joints of legs in a stance phase (see computation of τ_{stance}), all joints in the quadruped have proportional derivative (PD) controllers that are active at all times. While virtual forces provide many of the core functional aspects of the motions, e.g., using the stance legs to accelerate the front or rear of the body forwards or upwards, the joint control remains necessary to control the overall shape of the legs and body. Target angles are provided by predefined trajectories (head, neck, and tail), inverse kinematics (legs), or interpolated leg frame orientations (spine). The orientations of the head, the feet, and the leg frames are controlled with respect to a desired world-frame orientation, while all other joints servo to a desired local orientation, i.e., with respect to their parent link. The PD joint angle gains, k_p and k_d, are set to fixed values, i.e., they are not gait dependent.

Leg frames: The abstracted quadruped allows the controller design to be largely independent of the specifics details of the skeleton, such as the geometry and number of skeletal links used to model

(a) Walk.

(b) Trot.

(c) Pace.

(d) Canter.

(e) Transverse gallop.

(f) Rotary gallop.
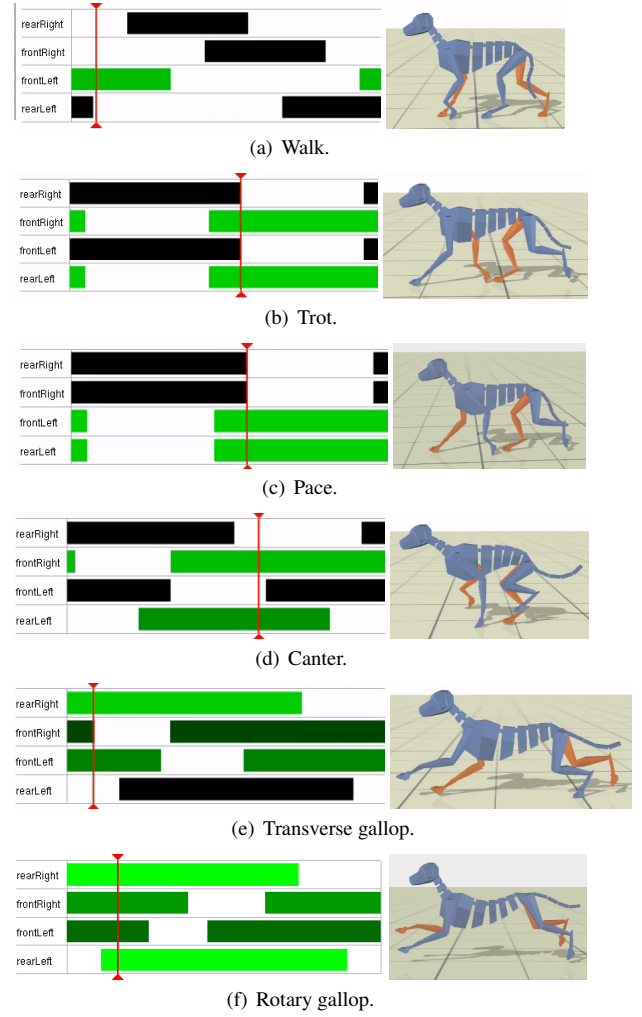
Figure 4: *Gait graphs. The red stripe indicates the current time or gait phase. Swing phases are drawn as solid bars. Currently active swing phases are colored green.*

specific body parts. The abstract quadruped model consists of front and rear *leg frames*, as shown in Figure 3. The frames are defined by the local coordinate frames of the articulated links of the spine to which the legs are attached. For the hind legs this corresponds the pelvis. The front legs are modeled as being directly attached to a link of the spine, which abstracts away the motion of the scapula. The height and orientation of the leg frames are key features whose motion is regulated by the controller. The leg frames are used as base links for all the virtual forces in the controller. We shall also use 'leg frame' to refer to the link together with its pair of legs where this can be done without introducing ambiguity.

Stance legs are largely responsible for controlling the motion of the leg frames. We first describe the control of the leg frame pitch, which is accomplished using the sum of torques applied to a given leg frame. Each leg frame receives applied torques from the joints that connect it to the stance leg(s), swing leg(s), and the neighboring spine segments, i.e., $\tau_{LF} = \tau_{stance} + \tau_{swing} + \tau_{spine}$. Here, τ_{stance} is the sum of all hip torques for the stance legs attached to a given leg frame, and τ_{swing} and τ_{spine} define analogous quantities for the swing legs and neighboring spine segments. Given a desired
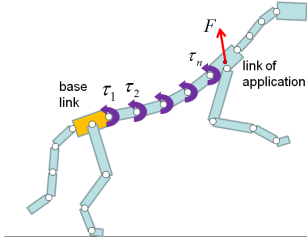
Figure 5: *Virtual forces as a control abstraction.*

Figure 6: *Virtual forces used in the controller.*

world-frame orientation for the leg frame and its current orientation, a desired value of τ_{LF} is computed with a PD-controller. In order to achieve the desired τ_{LF}, the positions of the hip joints are left uncontrolled, leaving τ_{stance} free to be computed according to $\tau_{stance} = \tau_{LF} - \tau_{swing} - \tau_{spine}$. This is analogous to the treatment of the stance hip in the SIMBICON framework [Yin et al. 2007]. If a leg frame has both of its legs in stance, the desired torque is shared equally among both stance joints. If the leg frame has both legs in swing, then τ_{LF} cannot be achieved and the leg frame orientation is no longer actively controlled. In order to engage the spine in creating full body motions, we can also choose to have the spine help in applying τ_{LF} to the leg frames. We currently apply this strategy to the front leg frame by requiring the spine to assume 50% of the required torque, while the stance legs assume the remainder. Without any engagement of the spine, we find that the torso exhibits a small wobble from side to side in some gaits. An equivalent strategy could be applied to the rear leg frame, although we have not explored this.

We next explain the virtual forces that are used to help guide the motion of the leg frames. First, a virtual force F_h is used to regulate the leg frame height to follow a height trajectory $h_{LF}(\Phi)$ using a PD controller according to $F_h = k_p e + k_d \dot{e}$, where $e = h_{LF}(\Phi) - h$. Second, a virtual force F_v is used to regulate the leg frame velocity according to $F_v = k_v(v_d - v)$. Third, a leg-specific virtual force $F_D(D)$ implements a phase-dependent force that is customised for each stance leg. This allows for modeling the individualized role of each leg in gaits such as the dog gallop [Walter and Carrier 2007]. Here, D measures forward progress in the gait and is computed as the ground-plane projection of $P_{LF} - P_{foot}$, where P_{LF} is the location of the origin of the leg frame, and P_{foot} is the location of the foot of a given leg. $F_D(D)$ initialized to zero and is tuned automatically through optimization (§4).

The virtual forces described above, i.e., F_h, Fv, and F_{Di}, are introduced to guide the motion of the leg frames, as illustrated with the dashed arrows in Figure 6. Achieving these virtual forces requires the use of the stance legs. This can be done in two ways: (1) the stance feet can be used as the base links to apply the desired applied forces on the leg frames; or (2) the leg frames can be used as base links which are then used to apply equal-and-opposite forces on the stance feet. However, the virtual force abstraction assumes that the base link acts as a stable anchor for the given chain of links. With this in mind, the mass and connectivity of the leg frames make them a better choice for base link than the stance feet. Because the forces F_h and F_v are not associated with any particular stance leg, they are shared equally across all stance legs. The net virtual force to be applied to a stance foot i is thus given by $F_i = -F_{Di} - F_h/n - F_v/n$, where n is the number of stance legs for the leg frame that stance leg i belongs to. If the leg frame has no stance legs, then no virtual force is applied, i.e., $F_i = 0$ for all legs belonging to that leg frame. The virtual forces that are realized for each leg are illustrated in Figure 6 with solid arrows. The thick colored lines illustrate the chain of links spanned from the base link, i.e., the leg frame, to the point
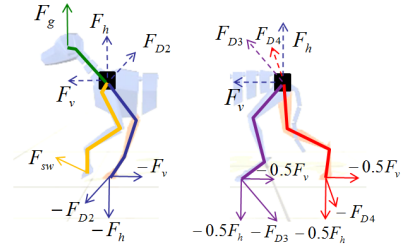
of application of the virtual force, i.e., the foot. Figure 6 also illustrates an example gravity compensation force, F_g, and an example swing leg force, F_{sw}. These will be elaborated shortly. It is also worthwhile noting that F_i is a net desired virtual force exerted by leg i; it is not the final net force seen by that leg. The actual forces that are transmitted through the leg remain unknown until after the equations of motion have been resolved for the current time step.

While virtual forces control the overall function of the stance legs, they leave their internal shape unconstrained. To retain control over the general shape of the multijointed leg, a target position is computed for the foot based on its current position on the ground and the desired height of the leg frame. Inverse kinematics (IK) is then used to compute target joint angles for PD-controllers at the joints. As illustrated in Figure 2, joint control torques and virtual force torques are simply summed.

Swing legs are controlled to follow desired **swing foot trajectories** from the toe-off location, P_1, to a target foot-strike location, P_2. Given the location of the foot, inverse kinematics is then used to compute target joint angles for the individual joints, which are then tracked using PD-controllers. The target location is computed using a velocity-based **foot placement model**: $P_2 = P_{LF} + (v - v_d)s_{fp}$, where P_{LF} is the default stepping location relative to the leg frame for each leg, and s_{fp} is a scale factor. These parameters are tuned in the gait optimization phase. Given P_1 and P_2, the foot height is defined by a trajectory, $h_{sw}(\phi)$, which is distinct for the front and rear legs. The target position in the ground plane follows a linear path between the two points according to $P = (1 - t_{sw})P_1 + t_{sw}P_2$, where t_{sw} advances with the swing phase of the given leg. While using $t_{sw} = \phi$ is adequate from a functional point of view, we introduce the flexibility to allow the swing legs to trail behind or advance forward more quickly by defining a piecewise linear function $t_{sw}(\phi)$. This is initialized to $t_{sw} = \phi$ and is refined during optimization to enable more natural swing leg trajectories.

Swing leg tracking force: In the absence of high PD-gains, the IK + PD-controller scheme described above does not provide sufficiently accurate foot tracking for high speed gaits with swing durations as short as $200ms$. As a more reliable alternative to simply increasing the PD-gains, an additional internal virtual force, F_{sw}, is introduced to pull the foot location towards its desired target location using a virtual spring and damper. The proportional gain for this foot tracking controller, k_{ft} follows a trajectory $k_{ft}(\phi)$. This trajectory is then tuned during the optimization phase. The joint-based PD-controllers are still retained in order to retain control over the shape of the multijointed leg. Early or late termination of swing may occur due to disturbances. An early foot strike proceeds on to a stance phase for swing phases $\phi > 0.8$. A late foot strike invokes a lowering of the target foot location by a fixed offset Δh.

Inverse kinematics: The same IK method is used for stance legs and swing legs. The solution is simple in nature because the world-relative foot pitch angle is known as a function of the leg swing

phase or stance phase. An analytic two-link IK solver then determines the position of the knee. The plane in which the IK chain acts is defined by a normal that is fixed in the leg-frame coordinate system, and the location of the relevant shoulder or hip joint.

Spine: Abstract modeling of the spine makes its control independent of the number of links used to model it. The leg frames each have desired world-frame orientations $\Omega(\Phi)$. The difference in orientation, as computed using quaternions, is divided evenly among the $n-1$ intermediate joints for the n-links that comprise the back, including the leg frames links. These joints are then driven to their target orientations using PD-controllers. The stiffness of the back is determined by the gains of the PD-controllers in the spine and is always held fixed.

Neck, Head, and Tail: The neck and head have associated pitch trajectories, $\Omega_{nh}(\Phi)$, modeled relative to the world frame. These are tuned automatically during optimization to minimize head wobble. The tail is controlled using local target angles at all joints. These target angles are held fixed over time. The base link of the tail is raised at higher speeds and the tail is also moved out of the way for sitting motions.

Gravity compensation removes the impact of gravity on the swing legs, neck, head, and spine. This is achieved using virtual forces, $F = -mg$, applied at the center of mass of the relevant links. For swing legs, the parent leg frames are used as base links for the virtual forces. For the neck and head, the closest grounded leg frame is used as the base link. This is usually the shoulders. Gravity compensation is also applied to the spine links. The virtual force is shared across the front and rear leg frames with respective weights of w and $1 - w$, where $w \in [0, 1]$ is the fractional distance along the spine of the given link, with $w = 1$ at the front leg frame and $w = 0$ at the rear leg frame.

Foot control adds toe-off and foot-strike anticipation to the gaits. This affects only the pitch of the foot, i.e., in the sagittal plane. Toe-off is modeled with a linear target trajectory towards a fixed toe-off target angle θ_a that is triggered Δt_a seconds in advance of the start of the swing phase, as dictated by the gait graph. Foot-strike anticipation is done in an analogous fashion with respect to the anticipated foot-strike time and defined by θ_b and Δt_b. The parameters $\theta_a, \theta_b, \Delta t_a, \Delta t_b$ are tuned automatically during the gait optimization phase.

Steering is implemented by changing the desired yaw angle of the front shoulders. This allows for moderate-speed turning behaviors, as the gaits are naturally robust to such variations. Faster changes in direction could possibly be achieved with the help of optimization or planning specific to such motions.

3.2 Gait Controller Summary

Feedback loops are implicitly or explictly embedded in the previously-described components in a number of ways. The feedback mechanisms are local in nature, i.e., taken individually, they do not require knowledge of the global state or the full equations of motion of the quadruped. The principal feedback mechanisms are: (1) The sagittal and coronal foot placement is adapted using the foot placement model. This is similar in nature to that used in prior quadruped work, e.g., [Raibert and Hodgins 1991]. (2) The velocity is further regulated using virtual forces at each leg frame, F_v. This is a type of virtual model control [Pratt et al. 2001]. (3) The leg frames have feedback paths for regulating height and pitch. The height is controlled with the virtual force, F_h. The pitch of the leg frames is controlled with respect to a desired world frame orientation, Ω_{LF}. This is done via the PD-controller that computes τ_{LF}, which is then implemented using a combination of the stance

hips and the spine. (4) Knowledge about vertical orientation is embedded in the PD-controllers that servo with respect to desired orientations in the world frame. This applies to the head-and-neck (Ω_{nh}), feet (θ_a, θ_b), and leg frames(Ω_{LF}). The target orientations are part of the set of parameters that are optimized offline. (5) Local joint feedback occurs in the PD controllers that help regularize the internal shape of the legs and the spine. (6) Early and late swing termination are sensed and reacted to. Early contact triggers a change to stance phase for the given leg, while a late foot strike triggers a fixed leg extension. (7) The gait is adapted as a function of the current velocity, as described in §4. For example, a galloping quadruped that is pulled back with an external force will revert to a canter, trot, or walk as needed.

Trajectories control many key components in the controller. These are summarized in Figure 2. The trajectories are modeled using piecewise linear segments and use 4–6 control points. The trajectories are all initialized to constant values and are then further tuned along with other parameters during an optimization phase, as will be described below.
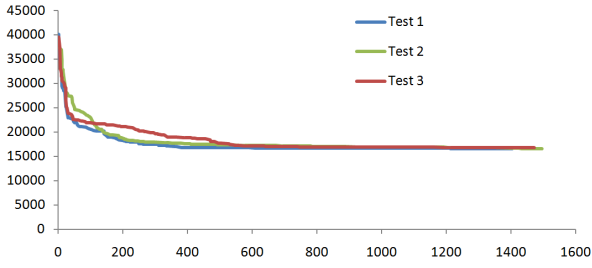
4 Gait Optimization

The control mechanisms and their default initialization allow for basic versions of quadruped gaits. We further use optimization to produce particular styles of gaits, including the tuning of gaits to more closely resemble the gaits of a dog. The free parameters include all the trajectory-based parameters; the stride duration; the step lengths for each leg; the foot placement feedback gain, s_{fp}; the foot control parameters, θ_a, θ_b, t_a, and t_b; and the gains used in specifying F_v and F_h for the leg frames. This comprises 144 parameters for asymmetric gaits (canter, gallop) and approximately half that for symmetric gaits (walk, trot). The complete list of parameters is provided in the supplemental material.

Motion capture data can play a very useful role in developing suitable trajectories and parameter settings for quadruped motions, although it is not a requirement. Animators have long used and studied reference video in the development of more natural motions, and our work is no exception in this regard. Our reference data consists of a set of 2D marker positions of the feet, shoulders, and hips that we have tracked from video [Abourachid et al. 2007] and which we correct for perspective and scale. The reference motions help in achieving more natural walk, trot, and gallop gaits, via the f_d term in the objective function to be described below. The canter, pace, parameterized leaps, sit, lie-down, and get-up motions are developed without the use of reference data.

The objective function to be minimized is defined as:

$$f_{obj}(P) = w_d f_d + w_v f_v + w_h f_h + w_r f_r$$

where f_d measures the deviation of the motion from available reference data (m), f_v measures the average deviation from the desired speed in both sagittal and coronal directions (m/s), f_h measures head accelerations (m/s^2), and f_r measures whole body rotations (degrees). These terms are weighted using $w_d = 100, w_v = 5, w_h = 0.5, w_r = 5$. This objective function is used for all gaits, although the initial parameter values, initial quadruped state, target velocities, gait graph, and reference data will be different for each specific gait. The f_d term is computed with the help of markers placed on the simulated dog in locations that approximate the locations for the available markers in the reference video data. f_d then penalizes a weighted sum-of-square distances between each corresponding pair of markers. We use markers on the feet, shoulder and hips, and top of the head. The foot tracking deviations were weighted less than the remaining distances by including an additional multiplicative factor of 0.4. The f_r term measures the

5

Figure 7: *Evolution of the objective function as a function of optimization iterations from three different initial parameter settings.*

difference between the desired heading and the actual heading of the character, as measured in degrees, by the $\arccos(x \cdot \hat{x})$, where x and $\hat{x}$ are the actual and desired forward pointing axes of a leg frame. This ensures that the quadruped runs forwards rather than sideways. The velocity error term, f_v, is defined as $||v - v_d||$, and encompasses both sagittal and coronal directions. v is the mean velocity as measured over a stride. The desired velocity in the coronal plane is zero. The desired velocity in the sagittal plane is an input parameter for the desired gait.

A greedy stochastic local optimization algorithm is used. On any given iteration, a new parameter vector P' is generated by perturbing the current best solution according to $P' = P + S \circ \Delta P$, where $\Delta P \sim U(P - 0.1R, P + 0.1R)$. $R = P_{\max} - P_{\min}$ defines the allowable range of parameter values. S is a parameter selection vector where each component of S is set to 1 with a 20% probability and 0 otherwise, and $\circ$ represents an entry-wise multiplication, i.e., the Hadamard product. If $f(P') < f(P)$ then P' replaces P as the current best solution. We speculate that many other choices of optimization method would also likely be successful. The success of a greedy optimization algorithm is indicative of the locally smooth nature of the optimization problem.

The robustness of the gaits is enforced by evaluating the objective over eight different scenarios for each parameter evaluation, with forces up to 350N applied for $0.1s$ on either the shoulders or hips during the second gait cycle. Between 7 and 15 locomotion cycles (approximately 6 seconds of simulation time) are tested for each evaluation, with higher speeds requiring more cycles. The initial state for each evaluation is obtained from the limit cycle of the current optimal motion. A gait at a specific desired speed is created using the f_v term in the optimization. When optimizing for the same gait at several speeds, we first optimize to create the lower speed gaits and then optimize for faster gaits in fixed increments.

In general we optimize a gait for a specific speed using $1,000 - 2,000$ evaluations and then increase the desired speed. This strategy is successful in beginning from an in-place canter or gallop to a full-speed canter or gallop. Optimizing a single gait takes between 2–10 hours in a single threaded implementation. Figure 7 illustrates the evolution of the objective function from three different initial paramater settings for a $1\,m/s$ trot. The initial parameter values for Test 2 and Test 3 are derived from those of Test 1 by adding an offset $\Delta p \sim U(-R/2, R/2)$, where U is the uniform distribution and R represents the allowed range for each variable. All three optimizations achieve very similar objective function values and produce visually similar gaits. The optimization makes significant alterations to the default initial values. For example, the height trajectory for the front and rear leg frames are initialized to constant values of $47cm$ and $45cm$, respectively and changed to $35 - 52cm$ within an optimized canter cycle. We attribute the success of the greedy optimization strategy to the generally well behaved nature
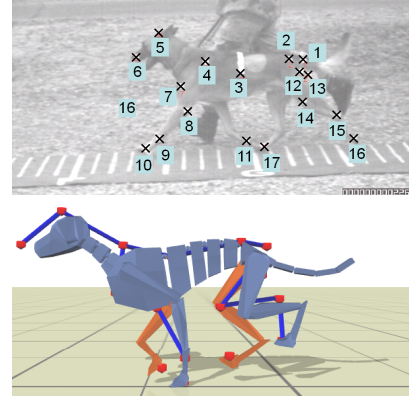
Figure 8: *Video data for capture of reference motions.*

of the control representation as well as an initial gait that is already functional in many respects.

Gait Selection and Transitions

The gait and gait parameters are selected as follows. Given the current speed, v and a goal speed, $\hat{v}$, a desired speed is computed at each time step according to $v_d = \min(\hat{v}, v + 0.5)$ if $\hat{v} > v$, and $v_d = \max(\hat{v}, v - 0.5)$ if $\hat{v} < v$. The gait and gait parameters that correspond to the desired speed, v_d, are then invoked. By construction, the desired speed is always within $0.5\,m/s$ of the current speed. The quadruped accelerates according to its capabilities rather than an explicitly specified rate of acceleration. Each gait is developed to work for a range of speeds: $0 - 1.5\,m/s$ for the walk; $1.0 - 2.5\,m/s$ for the trot; $2.0 - 3.5\,m/s$ for the canter, and $3.5 - 5.5\,m/s$ for the transverse gallop. A smooth transition between a slower gait and a faster gait is achieved by linear interpolation of all the gait parameters, including the gait graphs, over a specified transition range, $v_{\min} \leq v_d \leq v_{\max}$ according to $P = (1 - \alpha)P_{slow} + \alpha P_{fast}$, where $\alpha = (v_d - v_{\min})/(v_{\max} - v_{\min})$. This method is used for walk-trot transitions ($v_{\min} = 1.0\,m/s, v_{\max} = 1.5\,m/s$), and canter-tranverse-gallop transitions ($v_{\min} = 3.0\,m/s, v_{\max} = 3.5\,m/s$). An exception is the trot-to-canter transition, which we found did not interpolate well. A trot-to-canter transition is created with the help of a transition controller which is active for $0.3s$. This is automatically designed by optimizing an objective function that minimizes deviations from the desired speed during the transition and the first few canter gait cycles. The canter-to-trot transition is accomplished using a direct transition at the phase in the gait cycle where the gait graphs match, i.e., where the two gaits share the same stance legs. The trot-to-canter and canter-to-trot transitions happen at $2.2\,m/s$ in our implementation. In our experience, transitions are easily achieved if the gait graphs are similar, or if there are phases where the leg configurations match. If this is not the case, specialized transition controllers may be needed.

5 Parameterized Leaps

The same control abstractions used for periodic gaits can further be used to create skills such as jumps and leaps. We develop a flexibly parameterized leaping motion that can be used for leaping onto platforms, over obstacles, across gaps, and at given targets. Leaps are executed during the two phases in the trot cycle where the support is transitioning between the diagonal pairs. They are parameterized according to the known current speed and are defined by $\Gamma(h, d, v)$, where Γ defines a set of control parameters that produce a specific

jump, h is the maximum height of the leap, d is the distance traveled between takeoff and landing, and v is the speed at the start of the jump. We next describe the basic structure of a leap and the use of optimization to develop a parameterized jump.

A leap consists of 3 stages: loading, take-off, and airborne. The loading stage plants the shoulders and brings the hips under the body using gait patterns. The shoulders are also lowered in preparation for the jump using the desired height parameter. The stage ends when both back feet touch the ground. The take-off stage first stiffens the body. A vertical virtual force is then applied to launch the shoulders in the air. After a fixed elapsed time, a large virtual force is applied to the rear leg frame in order to launch the dog in the air. The stage ends when there is no contact with the ground. The airborne stage lasts until ground contact is reestablished with any leg. When jumping over a barrier, the quadruped is also made to retract its legs in order to provide additional clearance.

A landing controller is used when the y velocity of the center of mass becomes negative, i.e., the quadruped is now falling downwards. It uses the velocity foot placement model for all four legs in order to predict suitable target locations for the feet. If any target location is out of reach, it is moved to the be within reach at a given maximal leg length, which thus always yields a solvable IK problem and avoids leg hyper extension.

The 17 control parameters that define a leap, Γ, consist of the virtual forces and their durations, the timings of events in a stage, the degree of stiffening of the neck, spine, and back legs, and sagittal offsets for the desired position of back legs during the loading stage. A hand tuned leap is used as a seed point for the subsequent development of a parameterized leap.

Given the initial seed jump, an iterative stochastic procedure is developed with two goals in mind: (1) to generate leaps that sample a large portion of the (h, d, v) controller domain; and (2) for any given (h, d, v), to generate a minimal effort leap. The procedure builds a dictionary D of recorded leaps, $D : \{(\Gamma_i, h_i, d_i, v_i)\}$, initialized with the seed point outlined above. At each iteration, a leap $(\Gamma_j, h_j, d_j, v_j)$ is randomly selected from the dictionary and a variant of that leap is attempted. The variant is generated by perturbing v_j and Γ_j within given bounds. If the resulting leap fails or lands with an undesired body pitch, it is discarded. If the new leap is similar to an existing dictionary entry and the leap required less effort, that dictionary entry is replaced. Similarity is defined using a weighted Euclidean distance on (h, d, v), while the effort of a leap is defined as the integral of squared torques over the duration of the jump. We run this procedure overnight on a standard PC, which allows for thousands of iterations.

At run time, $\Gamma(h, d, v)$ is modeled using a nearest neighbor approach by employing the same distance metric as outlined above. The dictionary also serves as a model of the leaping capabilities at any point in time. A simple online planning algorithm scans the upcoming environment for discontinuities. The planner runs at the beginning of each gait cycle and assumes that the current speed is maintained until the leap is executed. The subset of leaps in the dictionary that satisfy $|v - \hat{v}| < 0.1$ is first identified, where v is the current velocity and $\hat{v}$ is the starting velocity of a leap in the dictionary. The key decisions to be made are the number of strides, n_s, to complete before executing the leap, and the choice of leap, i, as selected from the dictionary of available leaps. A planning horizon of 10 strides is used. For each discrete choice of n_s, $n_s \in [0, 10]$, the location of the start of the leap is estimated using the constant velocity assumption. From the subset of leaps available for the current velocity, the best leap is selected by minimizing $f_L(i) = |d - d_i| + |h - h_i| + \alpha E_i$, where d_i, h_i, and E_i are the distance, height, and effort of leap i in the dictionary. The effort is

measured as the sum-of-squared torques during the duration of the leap. The final selected plan is the one that minimizes f_L across all possible choices of n_s. A leap is executed when the best choice is to leap right away ($n_s = 0$). A hop-down controller is created by making small modifications to the controller for a small hop.

6 Sit, Lie-down, Stand-up, and Get-up

The design process for these controllers consists of first observing reference data (YouTube dog training videos) in order to establish approximate poses and timings for the motions. Key observed features were the step timing and transitions and the heights of the shoulders and hips. The motions are then modeled manually using the same gait graph framework used to control locomotion. When standing up from a sitting or a lying-down posture it is possible to transition to a standing posture, as described below, or to transition directly to a trot. Both are shown in the accompanying video.

Sitting is achieved in two phases with timed transitions. The first phase steps forward and slightly outwards with the back legs, one at a time, and then lowers the hips. The second phase decreases the desired hip height until the character is resting on its buttocks. To return to a standing pose, the quadruped steps forward with its front legs while lowering its shoulders slightly and increasing the hip height. The parameters involved in this motion are the sagittal and coronal step positions for the rear legs, timings, and the shoulder and hip height trajectories. The first attempt at designing the stand-up phase failed with the dog falling backwards. This was fixed with the help of an internal abstract virtual force that pulls the leg frames forward and by lowering the shoulders in order to generate some forward momentum. Sitting and return-to-standing requires approximately $1s$ each.

A lie-down action consists of two phases with timed transitions. The first phase is identical to the sitting module. A second phase steps forward with the front legs and lowers the shoulders until the quadruped is lying. To revert to a standing posture, a standing stage raises the hips and shoulders without stepping until the dog is standing. If the achieved standing pose is unstable, balance is rapidly restored by invoking one or two cycles of zero-speed walking. The parameters involved in this motion are the sagittal and coronal step positions for the rear and front legs, timings, shoulder and hip height trajectories, and shoulder twist and pitch trajectories. No virtual forces are used.

We develop a *get-up* controller that enables the dog to get up from a lateral decubitus position, i.e., lying on one side. The resulting motion is shown in Figure 13. When the dog has its feet under it, the trotting controller is extremely robust, and can recover gracefully from a wide range of scenarios. The strategy for *get up* is therefore to position the feet under the character so that the trotting controller can be engaged. This is accomplished in two phases. First, the spine is twisted while pulling in the front and rear legs to roll the dog onto its front feet. Then, once the shoulder-frame is vertical and the front feet are securely planted, the dog attempts to apply forward and upward vertical forces from the feet on the shoulder and hip frames. Since the front feet are in a more stable position, the shoulder force is approximately $3\times$ stronger. Once the shoulder frame is sufficiently high, the trotting controller is engaged and the desired hip height is slowly increased to normal standing height.

The design is challenging because some features of the framework may actively interfere with the goals of this controller. For example, the feet are not necessarily planted in a stable fashion and thus virtual forces may not work as desired. Most of the control is therefore implemented by modifying the dog's desired pose, particularly the desired relative orientation of hips and shoulder frames, and the desired end effector positions. We expect that the current manually
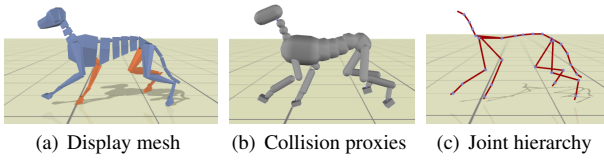
(a) Display mesh   (b) Collision proxies   (c) Joint hierarchy

Figure 9: *Quadruped construction.*

derived solution could be used as a seed point for further automatic tuning through optimization to arrive at potentially more robust and smoother motions.

7 Results

Implementation: Our model quadruped has the approximate dimensions of a female German shepherd dog. Figure 9 shows the display model, collision geometry and skeletal hierarchy. It has a shoulder height of $47cm$, a hip height of $45cm$, and a total mass of $34kg$. The articulated figure is comprised of 30 links: 4 links for each leg, 6 for the back, 4 for the tail, and 4 for the neck-and-head. There are 67 internal degrees of freedom: 7 per leg, 15 for the spine, 12 for the neck-and-head, and 12 for the tail. Open Dynamics Engine (ODE) is used to simulate the forward dynamics at $1000Hz$. We use the iterative solver in ODE. This allows for a simulation that is $3\times$ faster than real time when running on a current generation PC. During simulation we do not check for collisions between body parts, i.e., self collisions, in order to allow the legs and spine to be sufficiently flexible. The ground coefficient of friction is 1.0. We use torque limits of $100Nm$ in our gait robustness tests. We do not apply joint limits to the hips and shoulders because we found that joint limits in our simulator (ODE) introduced instabilities when combined with large ranges of motions. Joint limits are implemented on all other joints. In the supplemental material we include plots of the torques as a function of time for one leg of our model in order to show that they are generally well behaved.

Gaits and gait transitions: The gaits and their transitions are best seen in the video that accompanies this paper. We model six gaits and demonstrate transitions to and from these gaits: walk, trot, pace, canter, transverse gallop, and rotary gallop. Transitions can be made from a standing pose to any of walk, trot, or pace. Moving to a canter requires first passing through a trot. Moving to a transverse gallop requires first passing through a canter.

Optimization plays a crucial role in the design of the gaits. For controllers that are designed by hand it is particularly difficult to specify stepping patterns that use all the stance legs to propulse the body forward without the legs rotating about their vertical axes or slipping. We experimented with adding a *claw* friction model that assumed additional grip when pushing rearwards. However, the introduction of a leg-and-phase specific leg force $F_D(D)$ and the use of optimization to tune the gait eliminate the need to apply any special friction model.

Robustness: The resulting gaits are robust to pushes. Figure 10 shows an example reaction to a push disturbance. To quantify robustness, forces were applied at the front and back leg frames in a set of four directions (left, right, front, back), for a total of eight evaluations. The perturbations are applied at a single fixed point in time in the simulation. A gait is declared to be robust if it successfully recovers from all eight perturbations. We implement torque limits of $100\,Nm$ at every joint during robustness tests. The results are given in Table 1. In general, the shoulders can withstand larger perturbation forces than the hips. We speculate this is because more of the mass is concentrated there. The shoulders can take a force
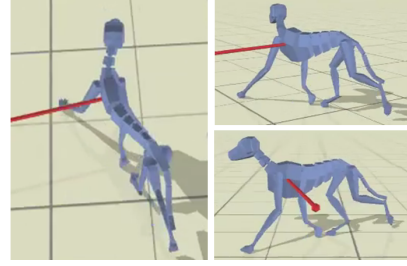
Figure 10: *Reaction to a 113 N push to the left applied for a duration of $0.65s$ to the front shoulders.*

| gait	$v\ (m/s)$	force (N)	duration (s)
walk	1.0	80	0.3
walk	1.0	230	0.1
trot	2.2	100	0.35
trot	2.2	200	0.1
canter	3.0	120	0.35
canter	3.0	400	0.1
t-gallop	5.0	100	0.35
t-gallop	5.0	400	0.1

Table 1: *Maximum omnidirectional force perturbations that the various gaits can recover from.*

almost twice as large, in the coronal direction, as compared to the hips. The difference is less pronounced for the sagittal perturbations. The walk is the least robust because the slower gait cycle means a longer wait until a foot placement can be enacted to help regain balance.

Variable terrain in the form of a series of low steps can be successfully traversed without anticipation or planning with any of the gaits, as shown in the accompanying video and in Figure 1 (middle). A leg becomes aware of an upcoming step at the beginning of its swing phase, which is when the height of the desired stepping location is obtained by querying the environment. No other terrain adaptation is performed in the controller.

Comparison to captured gait motions: We compare our simulated walk, trot, and transverse gallop gaits to captured dog motions. Objective measures of canine locomotion can be captured in a number of ways [Gillette and Angle 2008]. We obtain canine motion data from tracking of video data available to us [Abourachid et al. 2007]. Virtual markers are placed on the simulated quadruped to approximate the locations of the 17 markers that are tracked for the motion of the dog, as illustrated in Figure 8. We use Mn to denote marker n. The data is spatially aligned by matching the x positions of the pelvis as defined by M1. The captured data is corrected for perspective and scaled to compensate for differences in size. The height of the pelvis and the distance between the pelvis and shoulder are measured for both the simulated model and for the data. The y and x components of the data are then scaled by the ratio of these measurements. After the alignment and scaling, there still remain morphological differences between the simulated dog and the observed dog because the proportions of the simulated dog were designed independently.

A good correspondence is achieved for the foot placements (M10,M11,M16,M17). However, there remain significant differences between the simulated motion and the captured motion. Graphs of the relevant simulated and reference marker motions are included in the supplementary material. In walks and trots, the simulated shoulder (M4) and head (M5,M6) moves less than their cap-
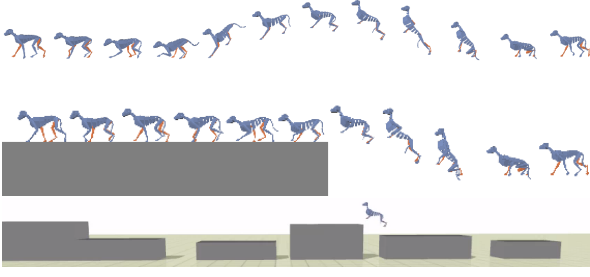
Figure 11: *Using parameterized leaps to traverse challenging terrain. Horizontal positions are not reflective of the horizontal motion. Top: detail for a leap across a gap; middle: detail for a leap up; bottom: profile view of terrain.*

tured counterparts. The head may move less in the simulation because of the f_{head} objective and the remaining difference in scale of the simulated dog. The markers on the shoulder and back of the dog may also not fully reflect the motion of the skeleton because they slide with the skin. The simulated knee (M14) and elbow (M8) move more than their captured counterparts. In comparing the simulated transverse gallop, the pelvis moves more (M2,M12), the shoulder moves less (M4), the head moves less (M6), and the head position is higher (M6). The simulated motion of the head is currently less dynamic than the data in part because of the stabilization objective of the optimization.

The simulated gaits are not yet capable of matching the speeds seen in real dogs of approximately comparable size. The simulated quadruped is functionally capable of trotting at up to 3 m/s, although speeds above 2.4 m/s begin to be less appealing and therefore the system transitions to canters at 2.2 m/s. The trot of the captured dog moves at 3.3 m/s. The simulated gallop travels at up to 5.7 m/s while a dog easily gallops at 6.5 m/s.

Role of spine: A variety of kinematic quadruped models often make use of a flexible spine [Skrba et al. 2008]. We hypothesize that the flexible spine also performs an important role for physically simulated gaits. To test this, we create a second dog model with a fused, rigid spine and develop an optimized 3 m/s canter gait for this new model. We then compare the resulting motion to that obtained for the flexible spine model for the same gait and speed. Both motions are shown in the accompanying video. The canter resulting for the rigid spine model exhibits considerably more oscillation in the pitch of the body and uses an overall lower body position. However, the specific results we obtain clearly do not preclude the possible existence of better results for a rigid spine model. Our principal observation is that the flexible spine appears to allow the back to stretch and compress during the motion while allowing the body to remains approximately horizontal throughout the stride.

Leaps and jumps: Quadrupeds can be highly agile creatures, as exemplified by the parkour skills of dogs [TreT 2011]. Our simulated quadruped can plan its way across a variety of challenging terrain in real time using the parameterized leaping coupled with the simple look-ahead planning strategy described in Section 5. It can leap up onto platforms, leap across gaps, leap over barriers, and leap at specific objects. Examples of this are shown in the video. Figure 11 shows an example terrain as well as the detailed poses of a leap across a gap and a leap up onto a platform.

The capabilities of the automatically synthesized space of leaps are defined by the leap distance, leap height, and the velocity of the trot at the start of the leap. Visualizations of the capabilities are given in the supplemental material. The synthesized dictionary of leaps provides approximate coverage over $h \in [0.7, 1.4]m, d \in$

$[0.2, 1.6]m, v \in [0, 1.7]$ m/s. Also evident is that longer distances can be achieved with higher initial speeds, while higher leaps are best done at low speeds. Interesting emergent behaviors are sometimes visible near the extremes of the capabilities. The quadruped may succeed with a jump onto a platform with only two or three out of the four legs and then eventually succeed in bringing the fourth leg onto the platform after some visible struggles.

The dictionary-based approach can potentially lead to neighbors in the (d, h, v) space that have significantly different parameter values, Γ. This is because there may be multiple ways of achieving the same goals. We construct a simpler parameterized model for $\Gamma(d, h, v)$ by applying principal component analysis (PCA) to the normalized (whitened) parameter vectors for a fixed velocity v and then build a linear model for the control parameter space defined by $\Gamma = P_0 + \alpha \Delta P_1 + \beta \Delta P_2$, where P_0 is the mean value of the parameters, ΔP_1, ΔP_2 are the first two principal components, and α, β are the independent parameters that define a given controller instance. This simple parameterization reliably covers the majority of the capability space achieved by the dictionary-based approach, as tested by constructing parameterized models for $v \in \{0.8, 1.0, 1.2\}$ for the trot gait. Subjectively speaking, the PCA-based parameterization of the leaps produces smoother and more graceful leaps, likely because of the smoothing that results from the PCA fit to the dictionary of leaps. With the exception of the leap across the $2m$ gap, the leaps shown in the final video use the PCA-based approach. The first two principal components are highly correlated with the height and distance of the jump. This is illustrated in the supplemental material. The PCA-based parameterization performs as well or better than using the full dictionary for most leaps, with the exception of the leaps that lie near the extremes of the quadruped's abilities.

The leaps and jumps produced by the controller remain robust to some degree of perturbation in the initial conditions. The leaps are typically successful if the quadruped is allowed to step at least two or three times with a constant velocity before the take-off, thereby allowing it to approach a limit cycle. If a jump is attempted prior to this point, it may lead to unintended distance and height, it may look unnatural, or in extreme cases it can cause the quadruped to trip and fall.

Sit, lie-down, stand, and get-up: Frames from a stand, sit, and lie-down animation are shown in Figure 12. The get-up motion is illustrated in Figure 13. Designing these controllers is currently done manually. We expect that more rapid design could be achieved with a suitable graphical user interface or the use of an optimization framework similar to that used to develop the other gaits and skills. The motions are robust to minor variations in the initial state of the quadruped. Larger variations require the use of the walk or trot controller in order to return the pose to a state that more closely resembles the initial standing pose used in the design of the controller. We have successfully experimented with transitioning from the sitting position to the trot gait without passing through an intermediate standing pose. Attempts to transition more directly from a sitting pose to a canter or transverse gallop were not successful, although we expect that it is possible to design such transitions. The lie down controller remains robust in a trial where the hind feet are on a raised 8 cm block.

Limitations: Our current model makes some significant approximations to the skeletal geometry of a dog. This is particularly evident in the modeling of the scapula, i.e., at the shoulders of our quadruped model. It is not clear what the ramifications of our simplifications are on the gaits and motions that can be achieved. For simplicity, the current implementation does not model self collisions. The motions of the head and tail are not modeled in detail. While we can can successfully demonstrate turning motions, we are
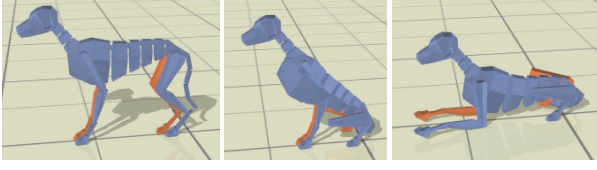
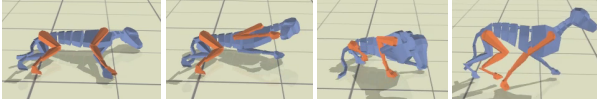**Figure 12:** *Quadruped stand, sit, and lie-down positions.*

Figure 13: *Getting up from a fall.*

not yet close to demonstrating the turning agility of most quadruped animals. The simple box geometry used to model the feet in our quadruped may be a limiting factor given that more agile motions may require foot models that exhibit compliance. The manual design of the controller for the get-up motion shown in Figure 13 was difficult and may indicate that we do not yet have the best representations and methods for that type of problem. It may be easier to design such motions using optimization.

8 Conclusions

In this paper, we have developed a skilled dynamically simulated quadruped, modeled on a dog. The quadruped is capable of a large variety of gaits, parameterized leaps and jumps, and sit, lie-down and stand-up motions. In support of these skills, we introduce the use of gait graphs, a dual leg frame model, a flexible spine, and a set of abstract forces tailored for quadruped motions. Optimization is applied to the control representation in order to define motions that satisfy given style or motion objectives. The robustness and range of capabilities are documented for the individual gaits and skills. The gaits are robust over moderate terrain variations with no explicit motion planning. The spine demonstrates flexibility while still being effective at transferring forces from the legs for propulsion.

There are many possible future directions to explore. Similar representations and methods should be applicable to modeling the motions of a potentially broad range of quadrupeds. However, there are still likely to be animal-specific details that require specific modeling, given the extremely broad range of gaits, behaviors, and skeletal dimensions of quadrupeds in the animal kingdom. Some of these motions and models may require more detailed biomechanical modeling of the musculoskeletal structure, including muscles and tendons, in order to produce simulated motions that have sufficient fidelity for some applications. This is also an element in the continued exploration of the connection between form and function in animal anatomy. Quadrupeds in nature further exhibit wide ranges of highly skilled behaviors that we do not model – our model is still unable to land on its feet like a falling cat, to scamper across a rock face like a mountain goat, or to pursue prey like a cheetah.

Appendix

Algorithm 1 Control Loop

1: **input** Φ: current stride phase
2: **input** dt: time step
3: **output** τ: vector of all joint torques
4: **output** Φ': updated stride phase
5: $\tau_{PD} = $ computePDTorques(Φ)
6: $\tau_{VF} = $ computeTorquesFromVirtualForces(Φ)
7: $\tau = \tau_{PD} + \tau_{VF}$
8: $\tau = $ applyNetLegFrameTorque(shoulders, τ)
9: $\tau = $ applyNetLegFrameTorque(hips, τ)
10: $\Phi' += dt/T$
11: **if** $\Phi' > 1$ **then**
12: $\quad \Phi' -= 1$
13: **end if**

Algorithm 2 applyNetLegFrameTorque(LF, τ)

1: **input** LF: leg frame
2: **input** N: number of stance legs in the leg frame
3: **input** τ: vector of all joint torques; current values
4: **output** τ: vector of all joint torques with modified τ_{stance_i}
5: $\tau_{LF} = $ getPDTorque(Ω_{LF})
6: **for all** LF stance leg i **do**
7: $\quad \tau_{stance_i} = (\tau_{LF} - \tau_{swing} - \tau_{spine})/N$
8: **end for**
9: return τ

References

ABOURACHID, A., HERBIN, M., HACKERT, R., MAES, L., AND MARTIN, V. 2007. Experimental study of coordination patterns during unsteady locomotion in mammals. *J. Experimental Biology 210*, 366–372.

ALEXANDER, R. 1984. The gaits of bipedal and quadrupedal animals. *The International Journal of Robotics Research 3*, 2, 49–59.

BLUMBERG, B., AND GALYEAN, T. 1995. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proc. ACM SIGGRAPH*, ACM, 47–54.

BUEHLER, M., PLAYTER, R., AND RAIBERT, M. 2005. Robots step outside. In *Int. Symp. Adaptive Motion of Animals and Machines (AMAM), Ilmenau, Germany*, 1–4.

COROS, S., BEAUDOIN, P., AND VAN DE PANNE, M. 2010. Generalized biped walking control. *ACM Transctions on Graphics 29*, 4, Article 130.

DE LASA, M., MORDATCH, I., AND HERTZMANN, A. 2010. Feature-based locomotion controllers. *ACM Transactions on Graphics (TOG) 29*, 4, Article 131.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proc. ACM SIGGRAPH*, 251–260.

FURUSHO, J., SANO, A., SAKAGUCHI, M., AND KOIZUMI, E. 2002. Realization of bounce gait in a quadruped robot with articular-joint-type legs. In *Proc. IEEE Intl Conf on Robotics and Automation*, vol. 1, 697–702.

GILLETTE, R. L., AND ANGLE, T. C. 2008. Recent developments in canine locomotor analysis: A review. *The Veterinary Journal 178*, 165–176.

GIRARD, M., AND MACIEJEWSKI, A. 1985. Computational modeling for the computer animation of legged figures. *ACM SIGGRAPH Computer Graphics 19*, 3, 263–270.

HECKER, C., RAABE, B., ENSLOW, R. W., DEWEESE, J., MAYNARD, J., AND VAN PROOIJEN, K. 2008. Real-time motion retargeting to highly varied user-created morphologies. *ACM Trans. on Graphics (Proc. SIGGRAPH) 27*, 3.

HERR, H., AND MCMAHON, T. 2001. A galloping horse model. *Intl Journal of Robotics Research 20*, 1, 26.

HODGINS, J., WOOTEN, W., BROGAN, D., AND O'BRIEN, J. 1995. Animating human athletics. In *Proc. ACM SIGGRAPH*, 71–78.

KIMURA, H., FUKUOKA, Y., AND COHEN, A. 2007. Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *Intl Journal of Robotics Research 26*, 5, 475.

KOHL, N., AND STONE, P. 2004. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proc. IEEE Intl Conf. on Robotics and Automation*, vol. 3, 2619–2624.

KOKKEVIS, E., METAXAS, D., AND BADLER, N. 1995. Autonomous Animation and Control of Four-Legged Animals. In *Proc. Graphics Interface*, 10–17.

KOLTER, J., RODGERS, M., AND NG, A. 2008. A control architecture for quadruped locomotion over rough terrain. In *Proc. IEEE Intl Conf. on Robotics and Automation*, 811–818.

KRASNY, D. P., AND ORIN, D. E. 2004. Generating high-speed dynamic running gaits in a quadruped robot using an evolutionary search. *IEEE Trans. on Systems, Man and Cybernetics 34*, 4, 1685–1696.

KRASNY, D., AND ORIN, D. 2006. A 3D galloping quadruped robot. *Climbing and Walking Robots*, 467–474.

KRY, P. G., REVERET, L., FAURE, F., AND CANI, M.-P. 2009. Modal locomotion: Animating virtual characters with natural vibrations. *Computer Graphics Forum 28*, 2.

LASZLO, J. F., VAN DE PANNE, M., AND FIUME, E. 1996. Limit cycle control and its application to the animation of balancing and walking. In *Proc. ACM SIGGRAPH*, 155–162.

LEE, Y., KIM, S., AND LEE, J. 2010. Data-driven biped control. *ACM Transactions on Graphics (TOG) 29*, 4, Article 129.

MARHEFKA, D., ORIN, D., SCHMIEDELER, J., AND WALDRON, K. 2003. Intelligent control of quadruped gallops. *IEEE/ASME Trans. on Mechatronics 8*, 4, 446–456.

MUICO, U., LEE, Y., POPOVIĆ, J., AND POPOVIĆ, Z. 2009. Contact-aware nonlinear control of dynamic characters. *ACM Transactions on Graphics (TOG) 28*, 3, 1–9.

PAUL, R. 1981. *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. The MIT Press.

PLAYTER, R., BUEHLER, M., AND RAIBERT, M. 2006. BigDog. In *Proc. of SPIE*, vol. 6230, 62302O.

POULAKAKIS, I., SMITH, J., AND BUEHLER, M. 2005. Modeling and experiments of untethered quadrupedal running with a bounding gait: The Scout II robot. *Intl Journal of Robotics Research 24*, 4, 239.

PRATT, J., CHEW, C., TORRES, A., DILWORTH, P., AND PRATT, G. 2001. Virtual model control: An intuitive approach for bipedal locomotion. *Int'l J. Robotics Research 20*, 2, 129.

RAIBERT, M. H., AND HODGINS, J. K. 1991. Animation of dynamic legged locomotion. In *Proc. ACM SIGGRAPH*, 349–358.

RAIBERT, M. H. 1986. *Legged Robots That Balance*. MIT Press.

RINGROSE, R. 1996. *Self-stabilizing running*. PhD thesis, MIT.

SKRBA, L., REVERET, L., HÉTROY, F., CANI, M., AND O'SULLIVAN, C. 2008. Quadruped animation. *Eurographics 2008-State of the Art Reports*, 1–17.

SOK, K. W., KIM, M., AND LEE, J. 2007. Simulating biped behaviors from human motion data. *ACM Trans. on Graphics (Proc. SIGGRAPH) 26*, 3, Article 107.

SUNADA, C., ARGAEZ, D., DUBOWSKY, S., AND MAVROIDIS, C. 1994. A coordinated jacobian transpose control for mobile multi-limbed robotic systems. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1910–1915.

TORKOS, N., AND VAN DE PANNE, M. 1998. Footprint-based quadruped motion synthesis. In *Proc. Graphics Interface*, 151–160.

TRET, 2011. Tret - parkour dog from ukraine, http://www.youtube.com/watch?v=pxelh_vm0uc. Accessed on January 9, 2011.

TSUJITA, K., TSUCHIYA, K., AND ONAT, A. 2001. Adaptive gait pattern control of a quadruped locomotion robot. In *Proc. IEEE Intelligent Robots and Systems*, vol. 4, 2318–2325.

VAN DE PANNE, M. 1996. Parameterized gait synthesis. *IEEE Computer Graphics and Applications 16*, 2 (March), 40–69.

VAN DEN BOGERT, A., SCHAMHARDT, H., AND CROWE, A. 1989. Simulation of quadrupedal locomotion using a rigid body model. *Journal of biomechanics 22*, 1, 33–41.

WALTER, R. M., AND CARRIER, D. R. 2007. Ground forces applied by galloping dogs. *The Journal of Experimental Biology 210*, 208–216.

WAMPLER, K., AND POPOVIĆ, Z. 2009. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG) 28*, 3, 1–8.

WONG, H., AND ORIN, D. 1995. Control of a quadruped standing jump over irregular terrain obstacles. *Autonomous Robots 1*, 2, 111–129.

WU, J., AND POPOVIĆ, Z. 2010. Terrain-adaptive bipedal locomotion control. *ACM Transactions on Graphics (TOG) 29*, 4, Article 72.

YE, Y., AND LIU, C. 2010. Optimal feedback control for character animation using an abstract model. *ACM Transactions on Graphics (TOG) 29*, 4, Article 74.

YIN, K., LOKEN, K., AND VAN DE PANNE, M. 2007. SIMBICON: Simple biped locomotion control. *ACM Trans. on Graphics (Proc. SIGGRAPH) 26*, 3, Article 105.

ZUCKER, M., BAGNELL, J., ATKESON, C., AND KUFFNER, J. 2010. An optimization approach to rough terrain locomotion. In *Proc. IEEE Intl Conf. on Robotics and Automation*, 3589–3595.

$\therefore$

5.8 Principal Geodesic Dynamics

Maxime Tournier, Lionel Reveret EG/SIGGRAPH Symposium on Computer Animation, SCA'12, Lausanne, Switzerland, july 2012.
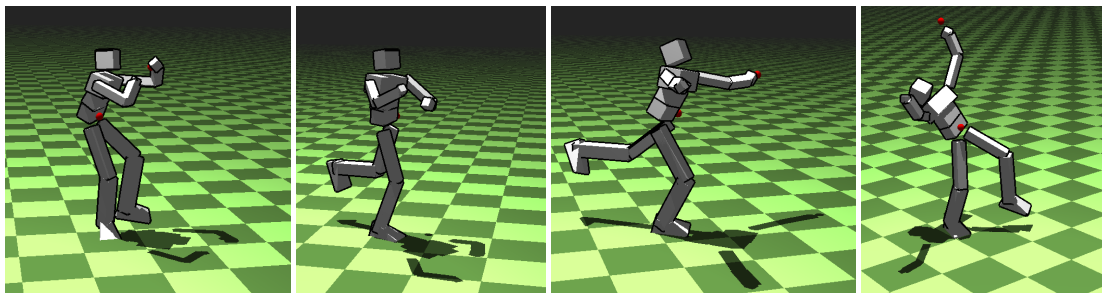
Principal Geodesic Dynamics

M. Tournier[1,2] and L. Reveret[3]

[1]INRIA, CNRS, Université de Montpellier 2
[2]DEMAR-LIRMM
[3]INRIA, CNRS, Université de Grenoble (LJK)

Figure 1: *One foot balance. 3 control features are present here: right foot, Center of Mass (CoM), and left hand (red). The system automatically adjusts the left leg position to maintain the CoM above the right foot while the user manipulates the left hand position.*

Abstract

This paper presents a new integration of a data-driven approach using dimension reduction and a physically-based simulation for real-time character animation. We exploit Lie group statistical analysis techniques (Principal Geodesic Analysis, PGA) to approximate the pose manifold of a motion capture sequence by a reduced set of pose geodesics. We integrate this kinematic parametrization into a physically-based animation approach of virtual characters, by using the PGA-reduced parametrization directly as generalized coordinates of a Lagrangian formulation of mechanics. In order to achieve real-time without sacrificing stability, we derive an explicit time integrator by approximating existing variational integrators. Finally, we test our approach in task-space motion control. By formulating both physical simulation and inverse kinematics time stepping schemes as two quadratic programs, we propose a features-based control algorithm that interpolates between the two metrics. This allows for an intuitive trade-off between realistic physical simulation and controllable kinematic manipulation.

Categories and Subject Descriptors (according to ACM CCS):   I.3.7 [Computer Graphics]: 3D Graphics and Realism—Animation

1. Introduction

While human body exhibits numerous degrees of freedom for producing motion, bio-mechanical studies showed they tend to be actuated in a highly-correlated way, resulting in similarly coordinated body motion. This fact has been typically exploited by recent successful approaches based on dimension reduction of human motion. This redundancy offers an opportunity to reduce the total complexity of animating a virtual character, by capturing this structure in a simple parametric model. It allows to generate better-looking animations at a lower computational cost. We propose in this paper an approach to use these reduced parameters directly at the core of a physically-based animation context by using them as generalized coordinates.

In the context of character animation, recent works have shown the potential of mixing dimension reduction with

physical simulation. What we propose here is a complete derivation of their integration into a coherent framework working in real-time and with stable control. Our physical simulation includes constraint-based unilateral ground contact model. Constraint-based model is favored over the penalty-based approach to avoid the risk of instability due too strong stiffness. Our approach allows to implement agile change of foot support during simulation, on user request, without planning the contact location explicitly. In order to maintain a real-time simulation while preserving important dynamic invariants, such as angular momentum, we used an explicit integrator based on variational geometric integrators.

To summarize, our contributions are the followings:

- an explicit time integrator for a reduced dimension pose model based on variational geometric integrators for articulated bodies,
- a simple quadratic programming control framework, providing trade-off between physical-simulation and kinematics control using a metric interpolation and real-time animation.

The remainder of the paper is organized as follow:

Section 2 reviews related works in the domain of dimension reduction in the context of both machine learning approach and physically-based animation. Recent works in the domain of task-space control are also discussed. Section 3 presents the algorithm for learning and approximating the pose manifold of a motion capture. Section 4 applies these ideas in the context of physically-based animation of characters using variational integrators. Section 5 on control develops the physical modeling to implement motion control. Examples of controllers for features tracking are presented in 6, leading to results for a balance controller. Finally, benefits and limitations are discussed in section 7.

2. Related Work

2.1. Dimension Reduction in Machine Learning

Recent advances in the machine learning theory allowed to design systems that capture geometric features as well as modeling time evolution on existing motion capture data. They allow synthesizing plausible motion while still retaining good control.

The idea of using machine learning to compute a higher-level parametrization of a motion given real examples has been widely explored over the past decade. [RCB98] used Radial Basis Functions (RBF) to learn an interpolation space from examples. [MK05] proposed to use geo-statistics to perform better interpolation in a control space. Brand and Hertzmann [BH00] propose a cross-entropy optimization framework to learn HMM for both structure and style from motion capture data. [GMHP04] cast the Inverse Kinematics (IK) problem as the optimization of the likelihood of

a Probability Density Function (PDF) over poses, knowing constraints. They propose to learn this PDF using a Gaussian Process Latent Variable Model, a generalization of PCA and RBF models that is computed on motion features (*e.g.* angles, velocities). Then, the corresponding pose likelihood function is optimized under geometric constraints.

All these methods exploit ever more complex non-linear statistical techniques to represent a statistical distribution over poses, possibly using regression against a feature space or directly optimizing the likelihood in which case the computational cost is high.

2.2. Dimension Reduction in Physically-based Animation

The benefits of dimension reduction have also been exploited in the context of physically-based animation, since they allow to reduce computational requirements while preserving most dynamic features of the full-dimension simulation. [PW89] pioneered the use of modal analysis in the context of computer graphics. The basic idea of this approach is to decompose a mechanical system with numerous, coupled mechanical degrees of freedom into an equivalent set of mechanically-independent one-dimensional degrees of freedom, usually called *modes*. The set of all modes forms the *modal basis* which can be seen as an alternate representation of the system DOFs.

Due to its numerous advantages, modal analysis has spawned a wide variety of research works ([JBP06, TLP06, WST09], among others) building on this basic principle. [KRFC09] proposed an extension of this technique to articulated rigid bodies: rather than performing the modal analysis on a mesh-supported, linearized FEM, the authors applied the modal decomposition to the dynamics of an articulated rigid body around a rest pose, obtaining new mechanically-independent angular degrees of freedom around a rest pose. This was the basis for animating locomotion behaviors as combinations of natural vibration modes. Simulation including contacts have been recently proposed in [JL11] and [NCNV*12]. However, in this case, modal analysis has the drawback that the joint stiffness must be known for the modal analysis to make sense. Instead, the reduced basis we obtained is not dependent on hand-chosen stiffness/damping parameters, but only on motion capture data. Besides, expressing the dynamics in our reduced pose parametrization is not restricted to small displacements.

Closer to data-driven statistical approach, [SHP04], and more recently [WMC11], present frameworks where statistical approaches and physically-based animation are integrated. Motion is parameterized with usual kinematics degrees of freedom. In their case, motion manifold learned from PCA acts as a constraint to plausible poses. Such an approach implies an optimization over a period time of several frames. We are proposing a different approach where latent

variables induced by manifold learning are directly used as generalized coordinates avoiding a costly optimization.

[YL08] present an alternate physically-based approach featuring dimension reduction for character animation. The goal of this work is to construct a data-driven basis of torques from motion capture data, then extract the coordinates corresponding to the *least* actuated torques. This *near-unactuated* basis is later used to synthesize upper-body, physically-based perturbations on top of existing motion capture animations. The main assumption is that the unactuated coordinates tend to be much more compliant, in the case of external disturbances, than the actuated ones. While this technique shows good results, its drawback compared to expressing the dynamics in the reduced dimension pose space is that it requires inverse dynamics, which is challenging to setup in the case of contacts, probably explaining that the motion perturbation is limited to the upper-body in their context.

2.3. Motion Control

The goal of this paper is not to propose new control algorithms, such as ones implementing walking. Our goal is rather to show how control can be implemented using our new formulation of the dynamics of articulated characters. We follow a *task-space control* approach as described in [AdSP07,MZS09,JYL09,LMH10,MLH10]. In this formulation, kinematic goals are formulated in some abstract *task*, or *feature* space, and the system solver is responsible for optimizing these goals using physics laws as a constraint *at each time-step*. This approach can be seen as a synthesis of the advantages of joint-space control and space-time optimization while mitigating their drawbacks: instead of solving a large, global optimization problem preventing interactivity, a smaller, easier problem is solved at each time-step. Besides, the burden of choosing and adapting individual control gains is shifted from the joint space to an abstract task-space.

In [LMH10, MLH10], the kinematic goals are expressed on accelerations, by forming a quadratic objective function. This objective is optimized under the dynamic equations and approximate contact constraints (no complementarity). A *feature* priority solving strategy is proposed, preventing certain control objectives to compete with each others. While this approach is very general and well-grounded for character control, it could arguably be improved especially by the use of a velocity/impulse formulation of dynamics and features, removing the need for second derivative which can be difficult to compute. Last, but not least, a reduced dimension motion model could improve the overall efficiency of this method significantly. These are the points we will address in the remaining of this paper.

3. Reduced Manifold for Kinematics

The configuration space of an articulated character, parameterized by its joint orientations, is not an Euclidean vector
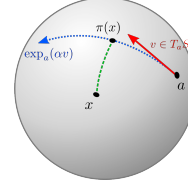
Figure 2: *The geodesic projection of point x over the geodesic $\exp_a(\alpha V)$ (blue) corresponds to the point where the geodesic segment in green has minimal length.*

space. This is the key limitation which prevents from applying safely standard analysis such as PCA. However, the pose manifold $SO(3)^n$ possesses a natural Riemannian structure, for which generalizations of Euclidean algorithms have been proposed, including multi-resolution analysis, interpolation, and statistics. Building on this property, [FLPJ04] proposed an extension of PCA to the case of Riemannian manifolds, first in the case of Lie groups having a compatible Riemannian structure, then on symmetric Riemannian manifolds.

Following [TWC*09] in the context of character animation, we have found the PGA algorithm to be a natural, simple dimension reduction technique, suitable for analyzing rotational pose data. We quickly recall main results in the context of character animation in order to introduce notations.

3.1. Geodesics on Lie Groups

Geodesics on a smooth manifold arise from a Riemannian *metric*, a smooth operator computing the norm of tangent vectors and the length of curves: geodesics are locally length-minimizing curves. Lie groups, on the other hand, are smooth manifolds where the group operations are smooth. A good introduction to these topics in the context of robot kinematics can be found in [MLS94].

In the case of a Lie group with a compatible Riemannian metric such as $SO(3)$, the metric and algebraic exponential coincide. Given such a Lie group G, the geodesic starting at $a \in G$ with initial tangent vector $V \in \mathfrak{g}$ (in body coordinates) is given by $\gamma_{a,V} = a.\exp(\alpha V)$, for $\alpha \in \mathbb{R}$ (*cf.* Figure 2). The geodesic distance between two points a and b of G is then given by:

$$\text{dist}(a,b) = \left\| \log(b^{-1}a) \right\|$$

In terms of Lie group operations, the projection of $x \in G$ on a geodesic starting at $a \in G$ with body tangent vector $V \in \mathfrak{g}$ is thus:

$$\pi_{a,V}(x) = a \exp(\alpha^\star V) \tag{1}$$

with

$$\alpha^\star = \underset{\alpha \in \mathbb{R}}{\text{argmin}} \; \text{dist}(x, a \exp(\alpha V))$$

3.2. Principal Geodesic Analysis

We now quickly outline the PGA algorithm for orientation data. More in-depth discussions can be found in [FLPJ04, SLHN10] for the general case. In essence, the PGA algorithm generalizes PCA using the geodesic distance and projection instead of the Euclidean ones. Given orientation samples, the first step is to compute their *intrinsic* mean $\mu \in SO(3)$, which minimizes the sum of geodesic distances to the samples [Moa02, Pen06]. The intrinsic mean provides a *coordinate-invariant* description of the data average. In contrast, an Euclidean mean of Euler angles or exponential maps depend on a particular choice of reference coordinate frame. Next, a set of mutually orthogonal geodesic directions is computed by maximizing the geodesic projections of the samples onto the corresponding geodesics.

Dimension reduction is obtained by selecting the $k \in \mathbb{N}$ most significant geodesic directions, where k is usually determined using a variance criterion. As the PGA algorithm only makes use of Lie group operations, its extension to the case of pose data, *i.e.* the product group $SO(3)^n$ with $n \geq 1$, is straightforward.

An approximate computation of the geodesic directions can be obtained by applying a PCA over the log-linearized samples at the intrinsic mean μ. While this approach somehow mitigates the advantages of PGA, we found it more practical to use and sufficient for animation purposes, while retaining the coordinate-invariance of the analysis. The differences between linearized and exact PGA are discussed in detail in [SLHN10].

We now describe how to apply PGA on motion capture data in order to construct reduced dimension skeleton kinematics.

3.3. Reduced Kinematics

We begin with an existing motion capture sequence represented as a set of m skeleton configurations $(g_i)_{i \leq m} \in G$ sampled over time. In practice, we found that breakdance sequences provide interesting variability in poses, with an associated set of 10 to 20 geodesics offering a range of motion large enough to allow generalization to other motion. Each configuration is a pair $(c, p) \in G = SE(3) \times SO(3)^n$, where $n \in \mathbb{N}$ is the number of joints, c is the rigid transformation describing the configuration of the root bone, and p contains the relative orientation of each joint with respect to its parent.

We will denote the pose space of relative joint orientations by $P = SO(3)^n$. We apply PGA on the pose samples $(p_i)_{i \leq m}$ to obtain the intrinsic mean $\mu \in P$, and a set of $k \in \mathbb{N}$ geodesic directions $V_j \in \mathfrak{p}$ describing the pose submanifold. With this decomposition, the new pose DOFs are k scalars $(\alpha_j)_{j \leq k} \in \mathbb{R}$ describing coordinates over the k principal geodesics. Practical computation of the intrinsic mean μ and geodesic directions V_j are detailed in [TWC*09].

We define the *reduced* pose parametrization as a smooth function $r : \mathbb{R}^k \to P$ that reconstructs a pose given k geodesic coordinates, $\alpha = (\alpha_j)_{j \leq k}$. Two mappings may be chosen for pose reconstruction, known as the canonical coordinates of the *first* and *second* kind [MLS94]. In practice, both have provided similar results for our purpose. We thus restrict to the first one:

$$r(\alpha) = \mu \, \exp\left(\sum_{j=1}^{k} \alpha_j . V_j \right)$$

We obtain the complete character forward kinematics, including the root DOFs, by composing r with the usual forward kinematics for articulated rigid bodies. We summarize this mapping as the following function:

$$f : SO(3) \times \mathbb{R}^3 \times \mathbb{R}^k \to SE(3)^{n+1} \tag{3}$$

where the configuration space $G := SO(3) \times \mathbb{R}^3 \times \mathbb{R}^k$ is a Lie group describing respectively the absolute orientation, absolute position and reduced pose coordinates of the character. f computes the absolute configuration of each of the $k+1$ bones of the skeleton.

The differential of f is obtained by the chain rule. Note, however, that the derivative of the exponential does not assume a simple formula as it does for the real scalar case, due to the non-commutativity of the rotation group. See [MLS94] for appropriate derivation.

4. Principal Geodesic Dynamics

Having described how to use a PGA-based reduced pose model in a kinematic animation context, we now move on to the physically-based animation of a character, using this reduced model as generalized coordinates. In this section we motivate and describe a time-stepping scheme for physically animating a virtual character parameterized using PGA. We derive a geometric integrator based on [KCD09]. Finally, we propose an approximated, explicit time-integrator, for use in a real-time simulation.

The use of reduced coordinates is natural in our context since we already have the reduced coordinates pose mapping as the PGA pose parametrization. In this case the constraint forces are implicit, and no constraint drift can occur by design. However, this formalism imposes the derivation of dedicated equations of motion, and notably to compute the reduced mass tensor and Coriolis forces according to the reduced coordinates mapping. In this context, the low-dimension of the PGA parametrization will be an advantage since it will keep the computational cost reasonable while still enforcing natural poses.

4.1. Geometric Integrator Derivation

We now derive the discrete equations of motion for a PGA-parametrized virtual character. For this, we use the geometric, variational integrator methodology presented in

[KCD09]. This work addressed the case of a single rigid body and consequently made the assumption of the left-invariance[†] of the Lagrangian. Instead, we derive equations for the general case of multiple articulated rigid bodies, where such an assumption does not hold. Unless stated otherwise, all the tangent maps and vectors used in the following are expressed in *body* coordinates, as opposed to *spatial* coordinates [MLS94].

Let $f : G \rightarrow SE(3)^{n+1}$ be the forward kinematics as defined in Equation (3), and $J(g)$ its Jacobian matrix. We assume that the inertia tensors for the $n+1$ bones composing the skeleton are given as a positive definite, block-diagonal matrix $\widehat{M}$ of dimension $6(n+1)$. The kinetic energy T is then defined as:

$$T(g,v) = \frac{1}{2} v^T J(g)^T \, \widehat{M} \, J(g) v$$

where $g \in G$ is the configuration, and the velocity $v \in \mathfrak{g}$ belongs to the Lie algebra. The above expression defines the generalized mass matrix, or inertia tensor as:

$$M(g) = J(g)^T \, \widehat{M} \, J(g)$$

The potential energy $V(g)$ is simply the gravitation potential.

Variational Integrator Assuming a Lagrangian $\mathcal{L} = T - V$ is given, the Hamilton equations of motion can be derived from the Hamilton-Pontryagin (HP) principle [KYT*06]. By properly discretizing the HP principle through a *quadrature*, one can obtain a symplectic integrator of arbitrary accuracy (called variational integrator) with excellent momentum and energy conservation properties. In the case of Lie groups, [KCD09] propose a first-order accurate quadrature of the HP principle in terms of a *group difference* map $\tau : G \rightarrow \mathfrak{g}$:

$$\delta \int_0^T \left(\mathcal{L}(g,v) + p^T (\dot{g} - v) \right) \, dt \quad \approx \quad (4)$$

$$\delta \sum_{k=0}^{N} h\mathcal{L}(g_k,v_{k+1}) + p_{k+1}^T \left[\tau\big(g_k^{-1}g_{k+1}\big) - h\,v_{k+1} \right] = 0$$

where the momentum $p^T \in \mathfrak{g}^*$ belongs to the Lie coalgebra, and h is the time step. In practice, we used the logarithm as the group difference map. The stationary conditions described by the above discretized HP principle result in a symplectic update rule for g, v and p, summarized in the following non-linear system (see Annexe A for more details):

$$\frac{\partial \mathcal{L}}{\partial v} (g_k,v_{k+1}) d\tau_{k+1} = p_k^T d\tau_k \, Ad_{d_k} + h\frac{\partial \mathcal{L}}{\partial g}(g_k,v_{k+1}) \quad (5a)$$

$$g_{k+1} = g_k \tau^{-1}(h\,v_{k+1}) \quad (5b)$$

$$p_{k+1}^T = \frac{\partial \mathcal{L}}{\partial v}(g_k,v_{k+1}) \quad (5c)$$

where $d_{k+1} = g_k^{-1}g_{k+1}$, $d\tau_k := d\tau(d_k^{-1})$ and Ad is the

---

adjoint[‡] representation of G. Forcing can be incorporated through an extended variational principle [KCD09]. If f_k is an external force applied at the kth time step, the integrated force $h\,f_k$ is added to the velocity update (5a):

$$\frac{\partial \mathcal{L}}{\partial v} (g_k,v_{k+1}) \, d\tau_{k+1} = p_k^T d\tau_k \, Ad_{d_k} \quad + \quad (6)$$

$$h\left(\frac{\partial \mathcal{L}}{\partial g} (g_k,v_{k+1}) + f_k^T \right)$$

Equations (5b) and (5c) are left unchanged.

4.2. Approximations for Real-Time Simulation

Two non-linear expressions in v_{k+1} complicate the integration update in equation (6). We propose two approximations to turn the update into a linear system, at the expense of symplecticity. Our experimental results show however that angular momentum is nearly conserved, which is not the case when using a non-Lie group integrator.

4.2.1. Quadratic Forces

The presence of the quadratic forces [MLS94] $\frac{\partial \mathcal{L}}{\partial g} (g_k,v_{k+1})$, containing Coriolis and centrifugal forces, is problematic since this term is quadratic in v_{k+1}. To avoid this issue, we approximate it by using the velocities from the previous time step, thus making it completely explicit:

$$\frac{\partial \mathcal{L}}{\partial g} (g_k,v_{k+1}) \approx \frac{\partial \mathcal{L}}{\partial g} (g_k,v_k)$$

In our simulator, we compute this term using central finite differences.

4.2.2. Difference Map Derivative

The second cause of non-linearity in the above variational update is the presence of $d\tau_{k+1}$ in the left-hand side of Equation (6), and this term depends non-linearly on v_{k+1}. As above, we approximate this term by reusing the velocity from the previous time step, which corresponds to $d_{k+1} \approx d_k$. Under these approximations, the velocity update becomes the following *linear* system:

$$\frac{\partial \mathcal{L}}{\partial v} (g_k,v_{k+1}) = \left[p_k^T \, d\tau_k \, Ad_{d_k} \quad + \quad (7) \right.$$

$$\left. h\left(\frac{\partial \mathcal{L}}{\partial g} (g_k,v_k) + f_k^T \right) \right] d\tau_k^{-1}$$

Note that if a more accurate *predictor* is available for v_{k+1}, the two approximations presented above can be adapted in a straightforward manner.

[†] *i.e.* constant in body coordinates

[‡] For a Lie group G, the adjoint $Ad_x : \mathfrak{g} \rightarrow \mathfrak{g}$ maps body coordinates to spatial coordinates corresponding to a tangent vector at $x \in G$.

4.3. Evaluation

We now evaluate the proposed integrator. To show the advantages of Lie group integration, we compare our integrator with one of the following form:

$$\frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) = p_k^T + h\left(\frac{\partial \mathcal{L}}{\partial g}(g_k, v_k) + f_k^T\right) \qquad (8)$$

Equation (8) is a naive adaptation of the Euclidean symplectic Euler integrator to the Lie group case, with explicit quadratic forces for the sake of comparison. The extra terms $d\tau_k Ad_{d_k}$ and $d\tau_k^{-1}$ in Equation (7) can be seen as appropriate change of frames for taking the Lie group *curvature* into account when summing external forces and previous momentum. A graphical interpretation of this can be found in [KCD09].

We compare the behavior of these two integrators on the following scenario: the character is in a gravity-less environment, we apply a short impulse on its right hand, along the positive Z axis. We choose a quite large time-step for an explicit integrator: $h = 0.1s$.

The accompanying video shows the trajectory of the Angular Momentum (AM) vector at the center of mass position. The Euclidean integrator clearly shows instability that our integrator does not exhibit, as seen in Figure 3. Another experiment is performed with a fall under gravity onto an infinite inclined slope. Again, the conservation of momentum appeared to be preserved by our integrator while some artifacts arose for the Euclidean integrator.
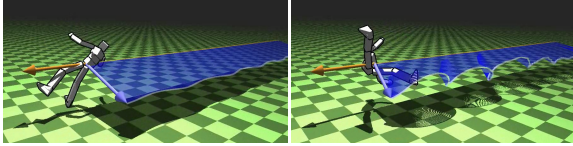
Figure 3: *The Angular Momentum* (blue) *is displayed after a short lateral impulse in a gravity-less environment. Our Lie group integrator* (left) *features a much better AM conservation than a non Lie-group one* (right).

5. Control Formulation

Now that a physically-based animation model has been derived, we turn to the problem of motion control.

5.1. Quadratic Programming

We describe a character control framework in the same spirit of [LMH10], but at the velocity/impulse level instead of force/acceleration. Equation (7) performs an explicit velocity update as a linear system, which can be summarized in a compact velocity-impulse formulation $Mv = p$. If we incorporate contact impulses, non-penetration and complementarity constraints into this system, we obtain the Karush-Kuhn-Tucker (KKT) conditions of a convex Quadratic Program (QP). We now formulate motion control by altering this QP.

The constrained dynamics of a character can be summarized as the following QP:

$$v_{k+1} = \underset{Av \geq b}{\operatorname{argmin}} \quad \frac{1}{2}v^T M v - p^T v \qquad (9)$$

where v is the velocity coordinates, p is the net momentum (*i.e.* the left-hand side in Equation (7)), M is the mass tensor, and A, b describe unilateral constraint geometry. We modify the original quadratic form, using a *control* quadratic form $\frac{1}{2}v^T Q v + c^T v$, in order to obtain a different behavior:

$$v_{k+1} = \underset{Av \geq b}{\operatorname{argmin}} \quad \frac{1}{2}\underbrace{v^T M v - p^T v}_{\text{dynamics}} + \underbrace{\frac{1}{2}v^T Q v + c^T v}_{\text{control}} \quad (10)$$

The control form may represent any *soft* kinematic constraint one wishes to satisfy on velocities, expressed as a quadratic form. We will see how this form is constructed in the next paragraphs. To enrich the user control, a simple interpolation parameter can be employed for the two quadratic terms to balance between a purely physical simulation, or better follow the kinematic constraints at the expense of physical realism.

One should note that modifying the original quadratic form in the above way effectively amounts to adding implicit forces of the form $Qv + c$, as can be seen on the corresponding KKT conditions. If the control form involves the root DOFs of the character, the associated forces *directly* act on the character root.

We now describe how to construct the control form in terms of control features.

5.2. Features Tracking Formulation

In practice, the control form is the aggregation of quadratic error terms, each one controlling a specific velocity-dependent *feature*. Let $F \simeq \mathbb{R}^d$ be an abstract *feature space* of dimension d. We define a *feature* γ as an affine map γ:

$$\gamma : \mathfrak{g} \to F$$
$$\gamma(v) = \Gamma v - \widetilde{\gamma}$$

We call $\Gamma \in \mathcal{M}_{d,n}$ the feature *matrix*, and $\widetilde{\gamma} \in F$ the feature *desired velocity*. Given a set of $m \in \mathbb{N}$ features $(\gamma_i)_{i \leq m}$, we build a quadratic form by simply summing all the feature squared norms:

$$\frac{1}{2}v^T Q v + c^T v := \frac{1}{2}\sum_{i=1}^{m} ||\gamma_i(v)||^2$$

More precisely, $Q = \sum_i \Gamma_i^T \Gamma_i$, $c = -\sum_i \Gamma_i^T \widetilde{\gamma}_i$, and unneeded constant terms are dropped. Features can be given

more importance in the final control form by *weighting* them accordingly. Such a weight is implicit in the definition of each feature, in the above formula.

We now illustrate the above formulation on a simple IK example. Let a function $e : G \to \mathbb{R}^3$ describe the error between an end-effector position and its desired position. The error for the current configuration g_k is $e_k := e(g_k)$. We obtain an estimation of e_{k+1} based on a first-order approximation at the current time step:

$$e_{k+1} \approx e_k + h\, J_{e_k}\, v$$

where J_{e_k} is the Jacobian matrix of e at g_k. We would like e_{k+1} to be as small as possible, thus we try to enforce:

$$J_{e_k}\, v = -\frac{e_k}{h}$$

The corresponding control feature is described by the pair $(J_{e_k}, e_k/h)$. In this example, we ask the error to be zero at the next time step, which corresponds to a *proportional* gain of $\phi_p = 1/h$. Smaller gains can also be provided, as well as *derivative* gains, in order to obtain a general Proportional-Derivative (PD) model for control features:

$$J_{e_k}\, v = -\phi_p\, e_k - \phi_d\, \dot{e}_k \qquad (11)$$

where $\dot{e}_k = J_{e_k} v_k$ is the error derivative for the current time step. As one would expect, the derivative gain ϕ_d tends to damp oscillations around the control target, at the expense of convergence speed. In the case where the error function is defined on a Lie group (*e.g.* when controlling orientation), the Lie group PD control described in [BM95] can be used instead. An example of the IK control feature is shown in Figure 4.

We now present some control results obtained using our system.

6. Examples of Controllers
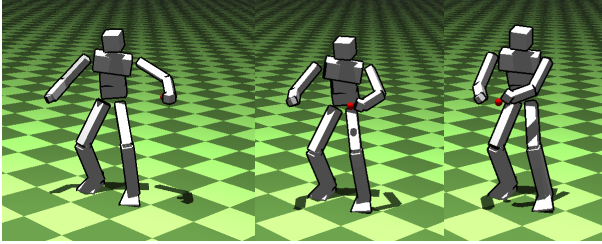
Figure 4: *Simple IK control feature. The desired position* (red) *is interactively manipulated by the user while the character maintains balance.*

6.1. Looking at Targets

Due to the full-body correlations captured by the PGA pose model, we have found the head rotational motion to be sometimes too important and thus penalizing for visual quality.

We thus implemented the following target tracking feature in order to gain an intuitive control on the head orientation.

The absolute head orientation is defined by a mapping $q : G \to SO(3)$. We compute the desired orientation as a direct orthogonal basis of $\mathbb{R}^3$ as follows:

- The first basis vector b_1 is the normalized view vector $w \in \mathbb{R}^3$, constructed as the difference between the view target and the head center
- The second basis vector b_2 is the projection of the world up vector $u \in \mathbb{R}^3$ onto the plane orthogonal to b_1, normalized
- The third basis vector b_3 is obtained by taking the cross-product of the two first vectors, finalizing the basis

Such construction is obviously ill-defined when the view and up vectors coincide. Should this situation happen, we simply skip this feature in the final control form.

Once the desired orientation $q^\star \in SO(3)$ is computed, we define an error function $e(g) = \log(q^{\star -1} q(g)) \in \mathbb{R}^3$ and obtain the corresponding control feature as described in 5.2. Figure 5 shows an example use of this feature control.
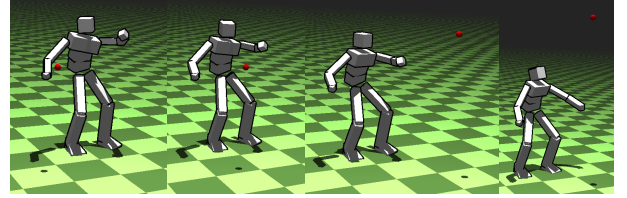
Figure 5: *An example of head orientation control: the character is looking at the red target.*

6.2. Balance Control

Let us now turn to balance-related feature control. As shown in [MZS09], the linear and angular momenta are two high-level quantities of particular interest in this context.

6.2.1. Center of Mass (CoM)

The CoM position is given by a function $c : G \to \mathbb{R}^3$ defined as:

$$c(g) = \sum_{i=1}^{n+1} m_i\, c_i(g)$$

where $n+1 \in \mathbb{N}$ is the number of rigid bodies in the skeleton, c_i computes the position of the CoM for the ith body, and m_i is the mass for the ith body.

Given a target position for c, an error function is easily derived from the above definition. Again, we obtain the corresponding velocity feature as described in 5.2.

6.2.2. Angular Momentum (AM)

In a feature-based control framework, it is possible to *directly* control the AM [LMH10], as opposed to an indirect AM control through the Zero Momentum Point [MZS09]. The AM with respect to the CoM c is defined as:

$$a = \sum_{i=1}^{n+1} R_i \, \mathcal{I}_i \, \omega_i \; + \; \sum_{i=1}^{n+1} m_i (c_i - c) \times \dot{c}_i \quad \in \mathbb{R}^3$$

where $R_i \in SO(3)$ is the absolute orientation of the ith body, $\mathcal{I}_i$ is the body-fixed inertia tensor of the ith bone, ω_i is the ith body-fixed angular velocity. The AM is linear in the rigid body velocities, which are themselves linear in the generalized velocities. Thus, the AM may be rewritten as a linear function of the generalized velocities:

$$a = H \, v$$

Thus, H is used as the AM feature matrix. We set the desired AM value to zero in order to prevent the character from undergoing destabilizing rotational motion.

Figures 6 and 7 compare the resulting behaviors when the AM control is turned off and on.
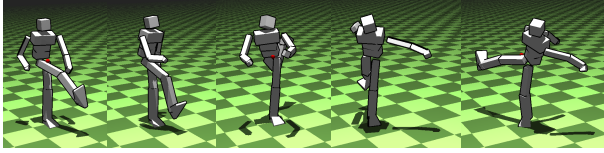
Figure 6: *One foot balance, without AM control: the character makes broad moves and quickly falls down. The red marker shows the desired CoM position.*
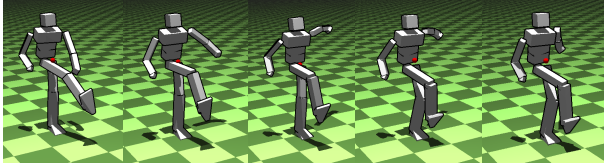
Figure 7: *One foot balance, with AM control: the character adjusts his arms and moves in a balanced way. The red marker shows the desired CoM position.*

6.2.3. Support Center

Instead of fully controlling the CoM position, it is usually sufficient to keep its projection on the ground *inside* the support polygon [JYL09, LMH10]. To do so, we record the position of the character contacts with the ground at each time step. We define the center of support $s \in \mathbb{R}^3$ as the mean of these contact points (*cf.* Figure 8). A control feature is added so that the CoM projection on the ground matches the center of support.
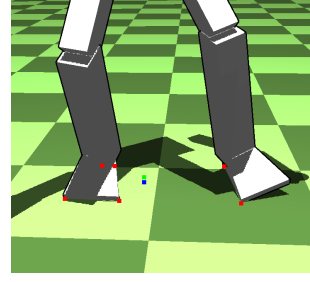

Figure 8: *The contact points* (red) *are used to define the support center* (blue) *as their mean. The CoM projection* (green) *is controlled by setting its desired value to the blue point.*

6.2.4. Results

Our balance controller uses the following feature set:

- Head and CoM projections should be at the support center,
- Feet should stay on the ground,
- Angular momentum should be minimal,
- CoM height.

We found the head projection criteria to be a simple yet effective way to maintain the character upright. Figure 1 and Figure 9 show examples of one-foot balance control. The accompanying video shows extended examples of interaction, with on-line change of support foot controlled by the user. This behavior is simply obtained by biasing the CoM and head projection targets toward the supporting foot position. The overall posture changes (standing, crouching, jumps) of the character are implemented through change in target height of the CoM.
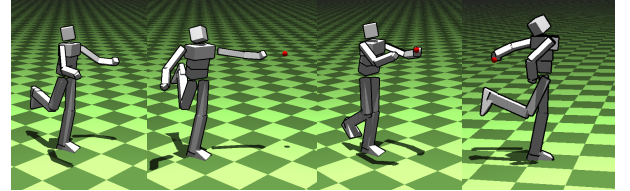
Figure 9: *One foot balance, while tracking an IK target.*

7. Discussion and Limitations

Efficiency As expected, the dimension reduction offered by the PGA allows to improve efficiency, since all the examples we proposed run in full real-time, at around 100Hz on a quad-core machine. We believe these performances to be improvable by engineering a better optimized code.

Joint Limits Though not described here for clarity, we have also added handling of joint limits. Our approach is based on a sampling of values observed in the learning joint orientation data. A Minimum Volume Enclosing Ellipsoid is

fitted to these data as a compact, analytical description of the allowed joint space. Unilateral constraints preventing the joint orientation from leaving this ellipsoid were incorporated in the control framework described in section 5. A conceptually similar, but more complex approach can be found in [HUH02].

Robustness Though the balance controllers we present performed generally well under user interaction, summing control terms in an unprioritized way will necessarily be subject to robustness problem in the case of a user insisting too much. We did not implement yet a prioritized optimization strategy as found in [LMH10], though it could have largely benefited from our reduced dimension strategy. This kind of framework produces much more robust controllers than simply summing objective terms, as the core balance objectives can not be perturbed by user IK requests. This could also improve the robustness of our controllers to impulsive perturbations.

Generalization Principal geodesic DOFs are completely dependent on the choice of the motion capture sequence used for learning. It thus questions the generalization capacity of such an approach. In our experiments, we mainly used the breakdance sequences for learning as shown in the accompanying video. It shows good properties for the interaction scenario we provide. The computation of PGA is extremely fast (less than a second in the example shown). The learning phase is not a limitation and different learning data could be tested on the fly if the resulting geodesics seem not appropriated to the type of motion edition wished by the user. More complex behavior such as locomotion is beyond the intended scope of this paper. As shown by numerous previous studies, simulation of locomotion requires careful controllers (typically finite state machines are needed). Our goal here was to firstly show that features-based controllers could be implemented in our Principal Geodesic Dynamics framework. The development of more complex controllers implementing locomotion is a natural extension of our work.

8. Conclusion

A Quadratic Programming control approach, exploiting a PGA reduced pose parametrization, has been presented. This approach allows to easily add small dynamic effects to the kinematic manipulation of a virtual character. We have experienced the system to be stable enough even when the contacts scenario is not known in advance. While such a control framework presented above can produce good visual results, it is not totally physically correct since the character is *externally* actuated through the features tracking. A more satisfying, but also more difficult approach, is to only use the *internal* actuators of the character to achieve a given task expressed using velocity features. Doing this is much more difficult from a theoretical point of view, as the true task-space control problem under unilateral constraints is non-convex, and thus much harder to solve. Some

algorithms have been proposed for solving such problems, such as a Sequential Quadratic Program. We did not try these methods yet, looking for real-time solutions in the first place. However, although these iterative algorithms are computationally-expensive, our reduced pose model may finally provide a low enough dimension reduction to make their real-time use possible.

Annexe A: Discrete Geometric Integrator

Let $d_{k+1} = g_k^{-1} g_{k+1}$. We now derive the stationary conditions for momentum, velocity and position in Equation (4).

Momentum Taking variations in p gives:

$$\sum_{k=0}^{N} \delta p_{k+1}^T \left(\tau(d_{k+1}) - h\, v_{k+1} \right) = 0$$

The *fundamental lemma of the calculus of variations* implies that for all $k \leq N$:

$$\delta p: \quad \tau(d_{k+1}) = h\, v_{k+1} \in \mathfrak{g} \qquad (12)$$

Velocity Taking variations in v gives:

$$\sum_{k=0}^{N} \left(\frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) - p_{k+1}^T \right) \delta v_{k+1} = 0$$

The same argument implies that for all $k \leq N$:

$$\delta v: \quad \frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) = p_{k+1}^T \in \mathfrak{g}^* \qquad (13)$$

Position Let us first remark that $\tau(x^{-1}) = -\tau(x)$, and let $d\tau_k := d\tau(d_k^{-1})$. Taking variations in g with fixed end-points ($\delta g_0 = \delta g_{N+1} = 0$) gives:

$$\sum_{0}^{N} h \frac{\partial \mathcal{L}}{\partial g}(g_k, v_{k+1}) \delta g_k - p_{k+1}^T d\tau_{k+1} \delta d_{k+1}^{-1} = 0 \qquad (14)$$

Noting that $\delta d_{k+1}^{-1} = -Ad_{d_{k+1}} \delta g_{k+1} + \delta g_k$, we split the sum by grouping terms in δg_k and δg_{k+1} together. After renumbering and exploiting the fixed endpoints property, we end up with the following equation for all $k \leq N$:

$$\delta g: \quad \frac{\partial \mathcal{L}}{\partial v}(g_k, v_{k+1}) d\tau_{k+1} = p_k^T d\tau_k\, Ad_{d_k} + h. \frac{\partial \mathcal{L}}{\partial g}(g_k, v_{k+1}) \qquad (15)$$

Acknowledgements

References

[AdSP07]   ABE Y., DA SILVA M., POPOVIĆ J.: Multiobjective control with frictional contacts. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 249–258. 3

[BH00]   BRAND M., HERTZMANN A.: Style machines. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 183–192. 2

[BM95]   BULLO F., MURRAY R. M.: Proportional derivative (pd) control on the euclidean group. In *In European Control Conference* (1995), pp. 1091–1097. 7

[FLPJ04]   FLETCHER P. T., LU C., PIZER S. M., JOSHI S. C.: Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging 23*, 8 (2004), 995. 3, 4

[GMHP04]   GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.:   Style-based inverse kinematics.   *ACM Trans. Graph. 23*, 3 (2004), 522–531. 2

[HUH02]   HERDA L., URTASUN R., HANSON A.: Automatic determination of shoulder joint limits using quaternion field boundaries. *In Proceedings of the 5th International Conference on Automatic Face and Gesture Recognition 2002* (2002), 95–100. 9

[JBP06]   JAMES D. L., BARBIČ J., PAI D. K.:   Precomputed acoustic transfer: output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. Graph. 25*, 3 (July 2006), 987–995. 2

[JL11]   JAIN S., LIU C. K.: Modal-space control for articulated characters. *ACM Trans. Graph. 30*, 5 (Oct. 2011), 118:1–118:12. 2

[JYL09]   JAIN S., YE Y., LIU C. K.: Optimization-based interactive motion synthesis. *ACM Trans. Graph. 28*, 1 (February 2009), 10–1. 3, 8

[KCD09]   KOBILAROV M., CRANE K., DESBRUN M.: Lie group integrators for animation and control of vehicles. *ACM Trans. Graph. 28*, 2 (May 2009), 16–1. 4, 5, 6

[KRFC09]   KRY P., REVÉRET L., FAURE F., CANI M.-P.: Modal locomotion: animating virtual characters with natural vibrations. *Comput. Graph. Forum 28*, 2 (avr 2009), 289–298. Special Issue: Eurographics 2009. 2

[KYT*06]   KHAREVYCH L., YANG W., TONG Y., KANSO E., MARSDEN J. E., SCHRÖDER P., DESBRUN M.:   Geometric, variational integrators for computer animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), SCA '06, Eurographics Association, pp. 43–51. 5

[LMH10]   LASA M. D., MORDATCH I., HERTZMANN A.: Feature-based locomotion controllers.   *ACM Transactions on Graphics 29number* (2010). 3, 6, 8, 9

[MK05]   MUKAI T., KURIYAMA S.: Geostatistical motion interpolation. *ACM Trans. Graph. 24*, 3 (July 2005), 1062–1070. 2

[MLH10]   MORDATCH I., LASA M. D., HERTZMANN A.: Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics 29*, 3 (2010). 3

[MLS94]   MURRAY R. M., LI Z., SASTRY S. S.: *A Mathematical Introduction to Robotic Manipulation*.   CRC Press, Inc., Boca Raton, FL, USA, 1994. 3, 4, 5

[Moa02]   MOAKHER M.:   Means and averaging in the group of rotations. *SIAM J. Matrix Anal. Appl. 24*, 1 (2002), 1–16. 4

[MZS09]   MACCHIETTO A., ZORDAN V., SHELTON C. R.: Momentum control for balance. *ACM Trans. Graph. 28*, 3 (2009), 1–8. 3, 7, 8

[NCNV*12]   NUNES R. F., CAVALCANTE-NETO J. B., VIDAL C. A., KRY P. G., ZORDAN V. B.:   Using natural vibrations to guide control for locomotion.   In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2012), I3D '12, ACM, pp. 87–94. 2

[Pen06]   PENNEC X.: Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision 25*, 1 (2006), 127. 4

[PW89]   PENTLAND A., WILLIAMS J.:   Good vibrations: modal dynamics for graphics and animation. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1989), SIGGRAPH '89, ACM, pp. 215–222. 2

[RCB98]   ROSE C., COHEN M. F., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl. 18*, 5 (1998), 32–40. 2

[SHP04]   SAFONOVA A., HODGINS J. K., POLLARD N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), SIGGRAPH '04, ACM, pp. 514–521. 2

[SLHN10]   SOMMER S., LAUZE F., HAUBERG S., NIELSEN M.: Manifold valued statistics, exact principal, geodesic analysis and the effect of linear, approximations. In *Proceedings of the 11th European conference on Computer vision: Part VI* (Berlin, Heidelberg, 2010), ECCV'10, Springer-Verlag, pp. 43–56. 4

[TLP06]   TREUILLE A., LEWIS A., POPOVIĆ Z.: Model reduction for real-time fluids. *ACM Transactions on Graphics 25*, 3 (July 2006), 826–834. 2

[TWC*09]   TOURNIER M., WU X., COURTY N., ARNAUD E., REVERET L.:   Motion compression using principal geodesic analysis.   *Computer Graphics Forum (EUROGRAPHICS'09)* (Mar. 2009). 3, 4

[WMC11]   WEI X., MIN J., CHAI J.: Physically valid statistical models for human motion generation. *ACM Trans. Graph. 30*, 3 (May 2011), 19:1–19:10. 2

[WST09]   WICKE M., STANTON M., TREUILLE A.:   Modular bases for fluid dynamics. *ACM Transactions on Graphics* (July 2009). 2

[YL08]   YE Y., LIU C. K.: Animating responsive characters with dynamic constraints in near-unactuated coordinates.   In *ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), SIGGRAPH Asia '08, ACM, pp. 112–1. 3

$$\therefore$$

5.9 CAGE-BASED MOTION RECOVERY USING MANIFOLD LEARNING

Estelle Duveau, Simon Courtemanche, Lionel Reveret, Edmond Boyer 3DIMPVT 2012 - Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, ZÃ$\frac{1}{4}$rich, Switzerland, oct 2012.

Cage-based Motion Recovery using Manifold Learning

Estelle Duveau   Simon Courtemanche   Lionel Reveret   Edmond Boyer

LJK, INRIA Grenoble Rhône-Alpes, France

Abstract

We present a flexible model-based approach for the recovery of parameterized motion from a sequence of 3D meshes without temporal coherence. Unlike previous model-based approaches using skeletons, we embed the deformation of a reference mesh template within a low polygonal representation of the mesh, namely the cage, using Green Coordinates. The advantage is a less constrained model that more robustly adapts to noisy observations while still providing structured motion information, as required by several applications. The cage is parameterized with a set of 3D features dedicated to the description of human morphology. This allows to formalize a novel representation of 3D meshed and articulated characters, the Oriented Quads Rigging (OQR). To regularize the tracking, the OQR space is subsequently constrained to plausible poses using manifold learning. Results are shown for sequences of meshes, with and without temporal coherence, obtained from multiple view videos preprocessed by visual hull. Motion recovery applications are illustrated with a motion transfer encoding and the extraction of trajectories of anatomical joints. Validation is performed on the HumanEva II database.

1. Introduction

Motion recovery using multiple view environments has known important developments in Computer Vision and Graphics. Applications to biomechanics and character animation require a reliable and interpretable measurement of motion. In that respect, the recovery of articulated motion appears as a relevant goal since the underlying skeleton provides a deterministic interpretation of motion. As a pre-process, it is now common to calibrate multiple views to obtain streams of 3D meshes from visual inputs such as silhouettes typically using space carving or visual hull [4]. The resulting information from visual hull is not temporally structured yet nor related to interpretable parameters in terms of motion. In contrast to existing methods for mesh tracking that use the coupling of an articulated skeleton and skinning to deform a known template, e.g. [20, 19], we propose here to relax the strong parametric constraint of the skeleton by making use of barycentric coordinates. In barycentric coordinates, a low polygonal version of the reference template, the *cage*, controls the deformation of the mesh by interpolation. We use Green Coordinates for their known properties of good conformal behavior [13]. The output is thus a sequence of temporally coherent coordinates of the vertices of the cage, controlling the deformation of a mesh template and constrained to lie on a pose manifold. To fulfill the goal of motion recovery, the cage has been carefully designed to follow a morphologically standardized representation of the body shape. It can thus be typically converted to an articulated skeleton representation.

For most methods, the accuracy of articulated tracking may still be challenged by lighting conditions, poor camera positioning or occlusions. One of the key factor is therefore the ability to bring an efficient regularization to the tracking problem. Latent spaces have been successfully used to constrain the manifold of plausible poses for human body tracking from monocular video or pose editing in animation. In this paper we investigate a similar strategy but in the novel context of motion recovery as a postprocess of 3D reconstruction of meshes from multiple view silhouettes. The direct link between the morphological cage and an articulated skeleton allows the use of available databases of motion capture data for the automatic learning of the manifold.

Our main contribution is thus the derivation of a mesh tracking algorithm, based on a Green Coordinates encoding, and regularized by a manifold learned whether by hand labeling of poses or automatically from motion capture data. We build on the formalism introduced by Prisacariu and Reid [14][15] for a rigorous derivation of the gradients of the object measurement with respect to the space of latent variables. We also provide the exact formulation of the gradients. We demonstrate our approach on different human performances : on temporally coherent clean meshes for consistency test [20], on the HumanEva II dataset [17] for objective evaluation and on non temporally coherent meshes obtained from a personal set-up adapted to visual hull for biometrics test.

1

2. Related Work

2.1. Motion recovery from a sequence of meshes

For mesh tracking, feature-level methods such as the patch-based approach of [2] offer a maximum flexibility. This approach is robust to large changes in the topology of the analyzed scene as no assumption is made on the internal structure but it does not fulfill the objective of motion recovery as no interpretable parametrization can be directly derived from this formulation. Following the parametrization encoded by our normalized cage, we deliver a more structured output. Furthermore, in case of very noisy input meshes, an approach that is too local risks providing unreliable results with non recoverable drift. This effect appears clearly on the HumanEva dataset [17], where the camera locations are not optimal for a preprocess step using visual hull reconstruction, leading to noisy input meshes.

Taking an opposite direction to feature-level approaches, the skeleton-based methods impose a strong prior on the parameter space. This approach has shown successful results such as in [20] and [8]. In both cases, a skeleton of a human or animal character has to be aligned on a reference template mesh. A skinning algorithm deforms the mesh, which is matched against 2D silhouettes. We propose here an alternative method for the deformation phase using Green Coordinates [13] and a 'cage' proxy to control the shape. The skinning algorithm is known to be sensitive to a careful tuning of vertices weights, where the automatic phase usually needs for manual adjustment. The vertices weights automatically provided by the Green Coordinates method show superior stability and do not require further tuning.

Tracking mesh data using a cage for barycentric-coordinates has first been proposed by [16] and recently improved by [18]. Unlike these works, we make no assumption on the temporal coherence of the input 3D data. Another major difference is in the regularization strategy. In [16], the cage deformation was constrained by a Laplacian smoothing. It has been reported that it did not prevent the cage from collapsing when tracking long sequences. This is our motivation for investigating a more robust regularization strategy using manifold learning.

2.2. Manifold learning for human motion tracking

The success of manifold learning for robust tracking of human motion from monocular videos has been typically proved with works such as [19] which uses GPLVM. For animation, an application of manifold learning has also been shown for 3D pose editing [9]. Other methods for animation editing using low dimensional subspace embedding exist. Among recent works, [3] uses different encodings for shape, pose and time. However, it requires an animation sequence as input whereas our learning samples are few and sparse. In our case, we therefore use a similar to [9] manifold learn-

ing approach, but in the novel context of cage-based animation from a sequence of meshes. We build on [14], who uses GPLVM to segment 2D images, by using a non-linear manifold as a deformation prior to track a cage-based animation. In particular, closed-form solutions of gradients are provided for a straightforward implementation of a simple optimization based on gradient descent.

The rest of the paper is organized as follows : in section 3, we describe how our template is defined and deformed. In section 4, we explain the learning procedure to obtain a latent space of coherent motion. In section 5, we detail latent space-based tracking. Section 6 shows results obtained with our method and evaluation.

3. Deformation Model

For our model-based approach, we need a shape model and a deformation model. Our shape model is a 3D mesh $\mathcal{M}$. To deform this mesh, we use a cage-based approach as it has better properties in terms of shape preservation and smoothness than skinning. In addition, it makes the motion parameters independent on the actual model as the same cage is used for every specific human model. Our template surface is defined as a mesh embedded in a coarse bounding mesh. The template mesh is called the *model*, the coarse bounding mesh is called the *cage*. The barycentric coordinates used to generate the model from the cage are given by the Green coordinates [13]. As mentioned in [13], unlike other barycentric coordinates, Green coordinates are empirically shown to be quasi-conformal in 3D. There are no requirements on the model : the only information used is the position of its vertices. Green Coordinates behave properly if there are no self-intersections in the cage when computing the coordinates. However, once the coordinates are computed, self-intersecting deformations of the cage do not create artefacts. Model vertices are expressed as a linear combination of cage vertices $\mathcal{V} = (v_1 \ldots v_n)$. Therefore a point P of the model is deformed according to :

$$P \mapsto \sum_{i=1}^{n} \phi_i(P)v_i + \sum_{j=1}^{m} \psi_j(P)n(t_j) \qquad (1)$$

where $(t_1 \ldots t_m)$ are the faces of the cage, $n(t_j)$ is the normal of face t_j and $\{\phi_i\}_{i=1\ldots n}$, $\{\psi_j\}_{j=1\ldots m}$ are the Green coordinates.

Being a coarse approximation of the model, the cage is easy to deform. However, it lacks an underlying structure to intuitively generate and deform it in a volume and shape-preserving way. Works such as [1] allow the user to interact with the cage using a small number of position and orientation constraints. However, how to generate the cage remains an issue. Inspired by skeletal joints, in order to provide a standard way of building and deforming the cage, we define oriented 3D quads located at main joints. These quads
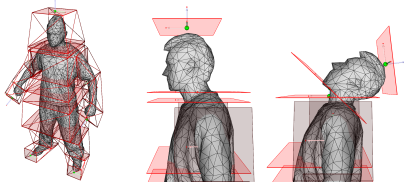
Figure 1: The Oriented Quads Rigging (OQR): template model and example of deformation.
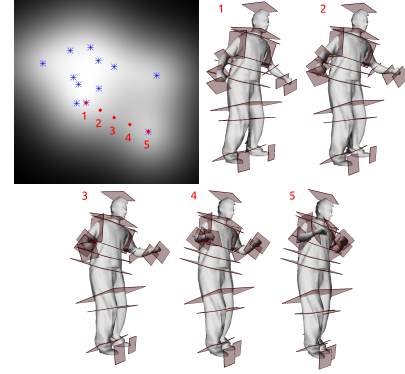

Figure 2: Example of 2D-latent space and corresponding deformations. The gray level corresponds to the variance at a given point in latent space. Blue crosses are the latent coordinates of the learned data-set. Red points and their corresponding reconstructions show that the latent space is suitable for interpolation between learned poses.

have 6 degrees of freedom (3 in translation, 3 in rotation) to enforce a structure. Their location corresponds to the interface between rigid areas of the model, intuitively, where joints would be located in joint-based animation.

Each oriented quad has a constant size which is not modified during the motion. The cage is then built procedurally from the vertices defined by the quads. Deforming the model then consists in setting the translation and orientation of each quad, which procedurally defines the vertices of the cage. The topology of the cage is computed once by defining relations between quads. The geometry of the cage is updated every time the OQR is modified with the positions of the vertices of the quads. For the sake of brevity, details could not be included in the present paper, which focuses on motion recovery. Finally, this cage defines the vertices of the model by linear combination of the vertices and normals of the cage (see Equation (1)).

As a result, the number of degrees of freedom of the deformation is greatly reduced. From 3 times the number of vertices of the model, we reduce it to 6 times the number of quads. In the example on Figure 1, it is reduced from 3×10002 to 6×21. This approach can be interpreted as an intermediate representation of an articulated character, between a hierarchical skeleton bind to a mesh using skinning and a standard non-hierarchical mesh. For the remainder of the paper, we refer to this representation as Oriented Quads Rigging (OQR).

4. Learning a manifold of deformations

Thanks to the constant size of the quads, some unrealistic deformations such as shrinking the circumference of an arm are avoided. However, nothing preserves the length of the limbs or non physiological configuration. The learning of a non-linear manifold of the possible positions and orientations of the quads allows to enforce such constraints. As in [14], where Gaussian Process Latent Variable Models are used to segment 2D images, we use GPLVM to learn a manifold of deformations. A Gaussian Process Latent Variable Model (GPLVM) is a low dimensional representation of the original data called the latent space. Each data point y_i of dimension d can be represented by a latent point x_i of dimension q, where q is smaller than d. Conversely, every

point x in the latent space projects to a data point through a Gaussian distribution defined by its mean μ and its variance σ^2. As derived in [11], for a data set of size n, we have :

$$P(Y|X,\theta) = \prod_{i=1}^{n} \mathcal{N}(y_i|0, K(\theta)) \qquad (2)$$

where $Y = [y_1 \dots y_n]$, $X = [x_1 \dots x_n]$ and $K_{i,j} = k(x_i, x_j)$ with k being the covariance function defined by the hyper-parameters θ. For instance, if k is a radial basis function, as in our case, it is written :

$$k(x_i, x_j) = \alpha exp(-\frac{\gamma}{2}(x_i - x_j)^T(x_i - x_j)) \qquad (3)$$

and the hyper-parameters are α and γ. To learn the GPLVM, Equation (2) is maximized with respect to both X and θ.

In our case, each data point y_i describes one deformation by the positions and orientations of the quads in the OQR. We align all the shapes with respect to the position and orientation of the quad at the pelvis. We then learn the latent variables and hyper-parameters by maximizing Equation (2). An example of a 2D-latent space can be seen on Figure 2. We impose back-constraints so that points close in data space stay close in latent space, which is not guaranteed in the usual formulation of GPLVM [12].

5. Tracking

To recover the motion, we must fit the model to the current representation of the shape, namely the *target*, which is, in our case, a 3D mesh. To do so, we derive a differentiable energy that measures how well the model fits the target, the model being the template mesh controlled by the
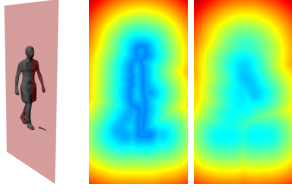
Figure 3: Example of distance fields - model and slicing plane (left), corresponding slice of signed distance field (right) as well as another slice in the sagittal plane.

cage. We differentiate this energy in the latent space to recover the deformation of the model. By doing so, we stay in the learned manifold i.e. in the space of plausible shapes.

5.1. Distance between two meshes

We make no assumptions on the temporal coherence of the target meshes. To measure how well the model mesh fits the target mesh, we thus need to measure the distance between two different meshes having potentially two different topologies. As we will minimize this distance, we need a function which is differentiable with respect to the model mesh. To that end we use a distance field built such that the target mesh is its zero level-set. Distance fields have the advantages of being easy and fast to compute. They are differentiable everywhere and can be used to measure the distance between meshes of different topologies.

Other approaches exist to deal with distances between non-corresponding meshes. One can first estimate vertex-to-vertex sparse correspondences, then use those pairs of vertices to compute the distance between the two meshes. However, computing correspondences is costly and can fail. Another approach is to directly use a mesh to mesh distance, such as the Hausdorff distance. However, such distances are often hard to compute and can not be differentiated because it implies finding optima. We thus use a distance field to avoid these issues [10]. Figure 3 shows examples of slices of distance fields that we obtain on noisy visual hulls.

Once the distance field is computed, we can define the distance function that has to be minimized. For a given mesh $\mathcal{M}$, the distance to the target $\mathcal{T}$ is computed as the sum of the distances of each vertex of $\mathcal{M}$ to $\mathcal{T}$. The distance of a vertex to $\mathcal{T}$ is the value of the distance field at the voxel to which the vertex belongs. Thus the distance function takes the following form :

$$d_{DF}(\mathcal{M}, \mathcal{T}) = \sum_{P_i \in \mathcal{M}} d_{DF}(P_i, \mathcal{T}) \qquad (4)$$

where $d_{DF}(P_i, \mathcal{T})$ is the value of the distance field generated from $\mathcal{T}$ at position P_i.

5.2. Shape Optimization

To recover the deformation of the model, we could minimize the distance between the model and the target in the unconstrained highly-dimensional OQR space. To limit the evolution of the model to plausible deformations, we differentiate this distance in the latent space. As a result, we stay in the low-dimensional learned manifold i.e. in the space of plausible poses.

We define an energy function to minimize with two terms. The first one measures how close the model and the target are. This is given by their distance as defined in Section 5.1. The second term takes advantage of the fact that at each point in the latent space, we can compute the variance which indicates how likely this latent point is to generate a valid data point. We use this variance as a regularization term to ensure that the shape generated from the latent point is likely to exist. We therefore try to find :

$$\hat{x} = \arg\min_x (d_{DF}(\mathcal{M}(x), \mathcal{T}) - \log(\sigma^2(x))) \qquad (5)$$

where x is the latent point, $\mathcal{M}(x)$ is the model generated from the latent point x and σ^2 is the variance at point x.

We do this by differentiating this energy with respect to the latent variables. This is done by using the chain rule :

$$\frac{d(d_{DF}(\mathcal{M}, \mathcal{T}))}{dx} = \frac{d(d_{DF}(\mathcal{M}, \mathcal{T}))}{d\mathcal{M}} \frac{d\mathcal{M}}{d\mathcal{V}} \frac{d\mathcal{V}}{dy} \frac{dy}{dx} \qquad (6)$$

where $\mathcal{V}$ is defined in Section 3 as the vertices of the cage and y is the deformed shape generated from x i.e. the set of positions and orientations of the oriented quads.

$\frac{d(d_{DF}(\mathcal{M}, \mathcal{T}))}{d\mathcal{M}}$ is computed by finite differences.

$\frac{d\mathcal{M}}{d\mathcal{V}}$ is computed by differentiating Equation (1) with respect to the vertices of the cage.

Each vertex in $\mathcal{V}$ is connected rigidly to one and only one oriented quad. $\frac{d\mathcal{V}}{dy}$ follows trivially.

y is the mean of the Gaussian distribution by which the latent point x projects to a data point :

$$y = Y^T K^{-1} K_x \qquad (7)$$

where K is the covariance in-between all the learned latent points $X = [x_1 \dots x_n]$ and K_x is the covariance between x and all the learned latent points. Differentiating Equation (7) gives :

$$\frac{dy}{dx} = Y^T K^{-1} \frac{dK_x}{dx} \qquad (8)$$

Each element of K_x is the covariance function k applied on x and one learned latent point. As a result, $\frac{dK_x}{dx}$ follows from Equation (3).

As for the regularization term, x projects to a data point with a mean as expressed in Equation (7) and a variance σ^2 such that :

$$\sigma^2 = k(x, x) - K_x^T K^{-1} K_x \qquad (9)$$

As a result :

$$\frac{d\sigma^2}{dx} = \frac{dk(x,x)}{dx} - 2K_x^T K^{-1} \frac{dK_x}{dx} \qquad (10)$$

Thanks to Equations (6) and (10), a simple first order method such as gradient descent can be used to minimize Equation (5) and recover the deformation of the model.

5.3. Pose Optimization

So far, we have captured the deformation of the shape as encoded in the latent space. As explained in Section 4, this shape is defined up to the global transformation Γ of the pelvis. As a result, we also need to recover the global translation and orientation of the model.

To do so, we minimize the energy without the regularization term by differentiating Equation (4) with respect to the pose parameters :

$$\frac{d(d_{DF}(\mathcal{M},\mathcal{T}))}{d\Gamma} = \frac{d(d_{DF}(\mathcal{M},\mathcal{T}))}{d\mathcal{M}} \frac{d\mathcal{M}}{d\Gamma} \qquad (11)$$

Pose and shape optimization can be done either jointly or iteratively until convergence. Joint optimization is more correct but converges more slowly as the search space is larger. Empirically, iterating between pose and shape optimization works faster and better, especially when tracking a sequence. Indeed, the rigid transformation and the deformation between two frames are small enough that optimizing first for the pose does not lead to a pose from which finding the good shape is impossible, and vice-versa.

6. Experiments and Discussion

We tested our method on three test sets. We first tested our approach on a temporally coherent mesh of high quality. We then tackled more challenging data by tracking a sequence of noisy visual hulls, without temporal coherence, learning the motion manifold automatically from the CMU Motion Capture Database (http://mocap.cs.cmu.edu/). This was first evaluated on the HumanEva II dataset [17]. We finally used it for gait analysis. All experiments have been done with the deformation model of Figure 1. It consists in a rigging of 21 oriented quads, which generates a cage of 86 vertices and 166 faces.

6.1. Evaluation on a temporally coherent sequence

We first tested our deformation optimization without the pose recovery. To do so, we use a bouncing sequence. The sequence of target meshes is made of a temporally coherent animated mesh of 10000 vertices and 20000 faces. We define the template mesh as one of these meshes simplified to 4000 vertices and 8000 faces. The training was done by manually locating the oriented quads on 16 frames of the sequence. We learn a 10-D manifold from these samples. Figure 4 shows the progression of the gradient descent towards
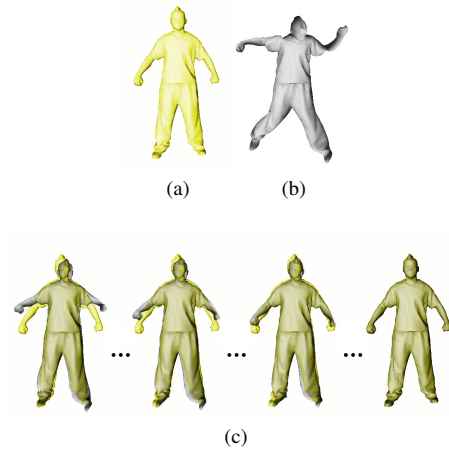

Figure 4: (a) Target model - (b) Initial pose - (c) Samples on the convergence trajectory.
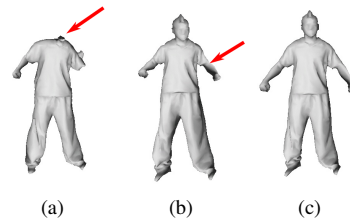

Figure 5: With the same target model and initial pose as in Figure 4 - (a) Result of the optimisation in the oriented quad space. Notice how the head collapses - (b) Result of the optimisation in the 10-D latent space without any regularisation term. Notice how the arms shorten - (c) Result with the regularisation term.

an optimal deformation. We can notice that, since we are optimizing in the latent space, each of the sample deformations is a valid deformation. Figure 5a shows the advantage of tracking in the manifold. As there is no structure on the cage, optimizing with respect to the oriented quads creates a shrinking of the cage as the quads are attracted to the closest patch of surface without any influence from the position and orientation of the other oriented quads. We can see on Figure 5 the influence of the regularization term of the energy. Without it, the model mesh is attracted to the shape that is the closest to the target mesh and can be generated from the latent space without any regard for the likelihood of this shape. Using the variance as a regularization term forces the optimal latent point to be more likely to generate a valid deformation.

As explained in Section 5.3, we also recover the pose. In practice, we noticed that pose recovery and deformation re-

Figure 6: Optimisation of both pose and deformation. The tracking (gray) is overlaid over the target (yellow).

covery done iteratively works better than joint optimization. Figure 6 shows results on the bouncing sequence.

6.2. Automatic learning on CMU database and validation on the HumanEva II dataset

Automatic learning : Instead of creating the training data by manually fitting the OQR to meshes, we also propose a method that uses existing databases of movements. From walking sequences in the CMU Motion Capture Database (http://mocap.cs.cmu.edu/), we automatically generate 120 training samples by rigidly mapping the quads of the OQR to the skeleton. It has to be done only once for a given frame of one of the subject's performance. After that, more samples can be automatically added to the training set using any available motion capture data samples for this subject. This training data is then used to build a 2D-manifold.

HumanEva II : We use the walk sequence of subject S4 in the HumanEva II dataset [17] for a quantitative evaluation. This data consists of 4 synchronized color cameras. Using the background subtraction software and the mixture of Gaussian distribution model provided with the dataset, we extracted the silhouettes from the videos. From these silhouettes, we generated the visual hulls using the Exact Polyhedral Visual Hull method [6]. As the scene was captured with 4 cameras facing each other, the visual hulls are a very coarse approximation of the observed shape as can be seen on Figure 7a. The model used for cage-based deformation is the mesh model for subject S4 generated by Stefano Corazza and the Stanford BioMotion Group using [4].

Local adjustment : One limitation of our method is its dependence on the training data. As with every manifold-based method, we are limited to what the manifold can reconstruct. To account for this limitation, we can first optimize with respect to latent variables to get as close as possible to the target mesh. Once this has converged, a few more iterations can be done with respect to the oriented quads to locally adjust the cage by differentiating the energy function with respect to the oriented quads :

$$\frac{d(d_{DF}(\mathcal{M}, \mathcal{T}))}{dy} = \frac{d(d_{DF}(\mathcal{M}, \mathcal{T}))}{d\mathcal{M}} \frac{d\mathcal{M}}{d\mathcal{V}} \frac{d\mathcal{V}}{dy} \quad (12)$$
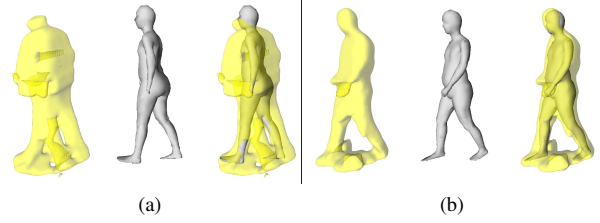

Figure 7: Examples of tracking on the HumanEva II dataset : (from left to right) visual hull, result and overlay of the model on the visual hull. (a) The location of the cameras can lead the visual hull to be a very coarse approximation of the observed shape. (b) When the visual hull quality increases, the tracking becomes more precise.
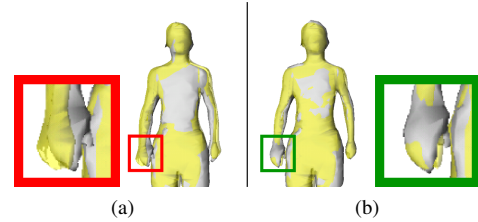

Figure 8: Example of local adjustment of the oriented quads - (a) before local adjustment - (b) after local adjustment.
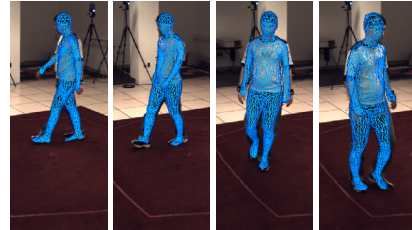

Figure 9: Reprojection on camera 3 of the tracked model for frames 5, 45, 102 and 148 of the sequence for subject S4 in the HumanEva II dataset.

An example of local adjustment is shown in Figure 8, where a 2D latent space does not capture enough variance to track the mesh properly. Table 1 shows the 3D error as specified in the HumanEva II dataset with and without the local adjustment for the first 150 frames. As our approach is not skeleton-based, the positions of the joints are computed by cage-based animation. On the first frame where motion capture data is available i.e. where the positions of the joints are known, the Green Coordinates are computed on these positions and then used to compute the new positions of the joints when the cage is deformed. With local adjustment, the mean error drops from 90mm to 38.6mm.

Figure 9 shows reprojections of the model tracked on the

Method	Frames	Mean error (mm)	Median error (mm)	Standard deviation (mm)
Corazza [5]	1-150	80.0		13.0
Gall [7]	2-1258	32.01		4.53
Our method - no local adjustment	1-150	90.0	87.3	24.1
Our method	1-150	38.6	39.2	4.1
Our method	1-300	85.2	55.8	55.8

Table 1: Error for sequence of subject S4 of the HumanEva II dataset. The mean, median and standard deviation in mm is measured for the first 150 frames without (third row) and with (fourth row) the local adjustment. The error increases after frame 150 (bottom row) as the quality of the visual hulls decreases (Figure 7a) to reach the levels of tracking without local adjustment.
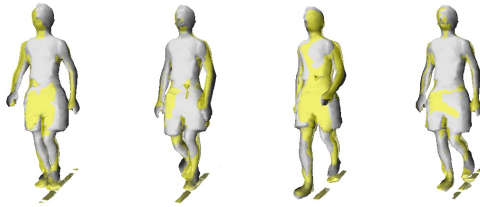

Figure 10: Pose and shape recovery - The tracking (gray) is overlaid over the target (yellow). The training data is built automatically from the CMU motion capture database and the target meshes are independent visual hulls.
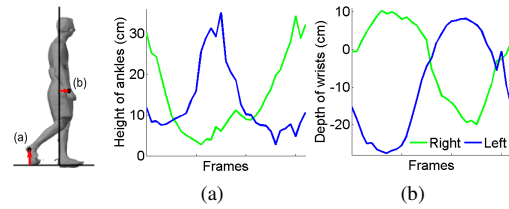

Figure 11: Application : motion analysis - (a) Absolute height of the right and left ankles - (b) Position relative to the pelvis in the sagittal plane of the right and left wrists.

sequence. We refer to the video for the whole sequence. For the whole walking sequence, the error is 85.2mm (Table 1). However, when considering only the first 150 frames, where the visual hulls are of better quality (Figure 7b), the error drops to 38.6mm. The error for the whole sequence is about the same as the error for the first 150 frames without local adjustment. This can be explained by the non-discriminative quality of the target mesh. Indeed, the coarseness of the visual hull (Figure 7a) prevents any fine adaptation of the shape by local adjustment. Table 1 also reports the results found in [5] and [7]. The method of Corazza et al. [5] also takes as input a series of visual hulls. The results are limited to the first 150 frames as re-initialisation is necessary after that due to the poor quality of the visual hulls. Thanks to the use of a manifold of deformations, even though the error increases after frame 150, we are still able to attain good results without re-initialisation. To our knowledge, Gall et al. [7] report the best results. However, they also use information not taken into consideration in this paper such as the appearance of the model.

6.3. Applications

Motion analysis: For gait analysis, walking and jogging sequences are selected in the CMU motion capture database

to learn a 10-D manifold for locomotion as explained in Section 6.2. The same subject is recorded performing walks and jogs at various speeds. Visual hulls are extracted from these sequences using [6]. The target meshes are visual hulls of about 7000 vertices and 15000 faces with holes and noise. We use one of those visual hulls cleaned and remeshed to 2500 vertices and 4500 faces as a model mesh. Examples of tracking are shown on Figure 10. From this sequence, we can extract locomotion cycles. Figure 11 shows the height of the feet and the swinging motion of the arms during a cycle. Because of the noise from the treadmill that can be seen on Figure 10, the trajectories of the feet are slightly noisier that those of the hands. We refer to the video for the results on varying speeds of walks and jogs.

Motion transfer : Once we have captured our parameterized performance, we can put any model we want in the cage to obtain a new animation (*motion transfer*). We can edit the original shape or put a texture on it as there is no drift in vertices position. This process is fully automatic as the computation of the Green Coordinates is automatic and gives smooth deformations. On the contrary, skinning usually requires user intervention to correct skinning weights, which makes the process of changing the model more tedious. Figure 12 shows an example of the transfer of the tracking shown in Figure 6 to a different 3D model.

Figure 12: Application : motion transfer from Figure 6.

7. Conclusion

In this paper, we tackle the problem of motion recovery from a sequence of 3D meshes. To this end, we have proposed a method using a cage-based deformation constrained by manifold learning. The use of a manifold regularizes the tracking problem, leading to a more robust solution. We use oriented quads as an underlying structure for cage-based animation. This gives an intuitive way to generate and deform cages, the Oriented Quads Rigging (OQR). On top of that, the use of cage-based animation allows for the tracking of any model, not just models with an intrinsic skeletal structure. Experimental results demonstrate the validity of our approach. We have evaluated our method on temporally coherent data and tested its usability by generating the training data from an existing motion capture database. However, our current method has not considered the fact that classic GPLVM struggles with learning a good model for complex datasets and becomes intractable for large datasets. Using a stochastic approach as in [21] may overcome these issues and lead to a better tracking on complex sequences.

References

[1] M. Ben-Chen, O. Weber, and C. Gotsman. Variational harmonic maps for space deformation. *ACM Trans. Graph.*, 28(3):34:1–34:11, July 2009. 2

[2] C. Cagniart, E. Boyer, and S. Ilic. Probabilistic deformable surface tracking from multiple videos. In *ECCV*, 2010. 2

[3] T. J. Cashman and K. Hormann. A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. *Computer Graphics Forum*, 31(2):735–744, 2012. Proceedings of Eurographics. 2

[4] S. Corazza, E. Gambaretto, L. Mundermann, and T. Andriacchi. Automatic generation of a subject-specific model for accurate markerless motion capture and biomechanical applications. *Biomedical Engineering, IEEE Transactions on*, 57(4):806 –812, april 2010. 1, 6

[5] S. Corazza, L. Mndermann, E. Gambaretto, G. Ferrigno, and T. Andriacchi. Markerless motion capture through visual hull, articulated icp and subject specific model generation. *International Journal of Computer Vision*, 87:156–169, 2010. 10.1007/s11263-009-0284-3. 7

[6] J.-S. Franco and E. Boyer. Exact polyhedral visual hulls. In *British Machine Vision Conference (BMVC'03)*, volume 1, pages 329–338, Norwich, Royaume-Uni, 2003. 6, 7

[7] J. Gall, B. Rosenhahn, T. Brox, and H.-P. Seidel. Optimization and filtering for human motion capture. *International Journal of Computer Vision*, 87:75–92, 2010. 10.1007/s11263-008-0173-1. 7

[8] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1746 –1753, june 2009. 2

[9] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. *ACM Trans. Graph.*, 23:522–531, Aug. 2004. 2

[10] M. Jones, J. Baerentzen, and M. Sramek. 3d distance fields: a survey of techniques and applications. *Visualization and Computer Graphics, IEEE Transactions on*, 12(4):581 –599, july-aug. 2006. 4

[11] N. D. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Machine Learning Research*, 6:1783 – 1816, 2005. 3

[12] N. D. Lawrence and J. Quiñonero Candela. Local distance preservation in the gp-lvm through back constraints. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 513–520, New York, NY, USA, 2006. ACM. 3

[13] Y. Lipman, D. Levin, and D. Cohen-Or. Green coordinates. *ACM Trans. Graph.*, 27:78:1–78:10, August 2008. 1, 2

[14] V. Prisacariu and I. Reid. Nonlinear shape manifolds as shape priors in level set segmentation and tracking. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2185 –2192, june 2011. 1, 2, 3

[15] V. A. Prisacariu and I. Reid. Shared shape spaces. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2587 –2594, nov. 2011. 1

[16] Y. Savoye and J.-S. Franco. Cage-based Tracking for Performance Animation. In *ACCV'10 :the Tenth Asian Conference on Computer Vision*, Queenstown, New Zealand, 2010. 2

[17] L. Sigal, A. O. Balan, and M. J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *Int. J. Comput. Vision*, 87(1-2):4–27, Mar. 2010. 1, 2, 5, 6

[18] J.-M. Thiery, J. Tierny, and T. Boubekeur. Cager: From 3d performance capture to cage-based representation. In *ACM SIGGRAPH 2012 - Talk Program*, 2012. 2

[19] R. Urtasun, D. J. Fleet, and P. Fua. 3d people tracking with gaussian process dynamical models. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, pages 238–245, Washington, DC, USA, 2006. IEEE Computer Society. 1, 2

[20] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3):97, 2008. 1, 2

[21] A. Yao, J. Gall, L. van Gool, and R. Urtasun. Learning probabilistic non-linear latent variable models for tracking complex activities. In *Proceedings of Neural Information Processing Systems (NIPS)*, Granada, Spain, December 2011. 8

CHAPITRE 6

Conclusion et Perspectives

Parmi les contributions informatiques les plus aboutis de ce travail, une des plus importante est la méthode de suivi vidéo de maillage 3D par apprentissage. Elle fait le lien entre une souplesse de représentation de la forme externe et une description anatomique du mouvement. Ce lien se traduit, d'une part, par la possibilité de réaliser un premier apprentissage automatique de la forme externe à partir de données de mouvement articulé, via le paramétrage OQR. D'autre part, la cohérence temporelle du maillage suivi permet de générer un mouvement squelettique 3D, via une stratégie de cinématique inverse pondérée. Pour le mouvement humain, cette méthode a donné de bons résultats chiffrés sur la base de données de référence HUMAN-EVA. Dans le cadre du projet ANR MORPHO auquel je participe pour évaluer l'utilisation des données de maillages 3D en mouvement dans un contexte de biomécanique, une validation plus poussée sera menée sur une base de données similaire à HUMAN-EVA mais plus complète. Dans le cas des rongeurs, cette approche de suivi sera aussi évaluée et développée plus avant à travers la collaboration avec l'Institut Clinique de la Souris (ICS), au sein de l'Institut de Génétique et Biologie Moléculaire et Cellulaire (IGBMC) à Strasbourg. Un système multicaméra est déjà en place au sein de l'ICS et des tests intensifs en nombre d'individus testés vont être menés pour accroître la robustesse et l'applicabilité de la méthode.

Les travaux en modélisation physique ont permis d'aborder des méthodes de la mécanique des solides articulés. Pour l'instant, nos contributions dans le domaine physique ont porté sur des constructions logicielles focalisées sur un objectif particulier afin d'en maîtriser tous les aspects. La perspective d'un passage à des logiciels plus complexes comme SOFA (INRIA) ou OpenSim (Université de Stanford) permettra d'aborder des problèmes intégrant plus d'aspect de modélisation (tissus mous) et d'avoir un impact plus large dans la communauté scientifique par l'effort de standardisation que ces logiciels représentent. Leur potentiel de modélisation pour aider à l'analyse sans marqueurs du mouvement anatomique est un domaine à découvrir. Pour

l'instant, je n'ai considéré que les structures squelettiques et la surface externe. La modélisation musculaire est l'étape suivante évidente.

Outre l'implantation à l'ICS, les efforts expérimentaux ont aboutis avec le projet d'équipement KINOVIS à Grenoble. Deux salles dédiées à l'analyse du mouvement vont être construites : une dans les locaux de l'INRIA à Montbonnot où une cinquantaine de caméras seront connectées pour permettre un large champ de capture de maillage 3D en mouvement (10mx10m) et une à la faculté de Médecine de Grenoble, au laboratoire d'Anatomie (LADAF), où 8 caméras seront couplées à une cinéradiographie biplanaire. En plus du travail sur les rongeurs, j'explorerai un nouveau domaine à l'aide de cette plateforme, l'orthopédie de la main chez l'homme. Le volume couvert et les méthodes d'analyse pour l'étude des rongeurs sont techniquement comparables à ce qui est nécessaire pour les mains. Comme pour les rongeurs, le but sera de mettre en relation des données squelettiques internes et des mesures dynamiques de surface par vidéo. A terme, l'idée est de fournir des outils de diagnostic se passant des rayons X, qui auront servis à l'apprentissage et à la validation.

Les perspectives dressées ici sembleraient s'éloigner un peu plus de l'Animation 3D en se rapprochant de la Biomécanique. En ce qui concerne mes activités présentes et futures, je pense au contraire que les deux se retrouvent en cherchant à faire de la Capture de mouvement un objet scientifique en soi. L'ancrage dans la mesure permet de garder un oeil sur la réalité et non pas sur la manière dont on veut la voir. Ma formation initiale à la recherche continuera à aller vers cette approche métrologique du mouvement. Sans contradiction avec cette démarche, il est tout à fait envisageable d'accompagner un modèle validé d'animal ou d'humain à trouver sa place en production 3D. Pour les prochaines années, la mise en place de nouveaux outils expérimentaux offrent des perspectives inégalées de mesure de mouvement et stimuleront la recherche de nouveaux modèles fiables.

Bibliographie

[ABdLH13] Mazen Al Borno, Martin de Lasa, and Aaron Hertzmann, *Trajectory optimization for full-body movements with complex contacts*, IEEE Transactions on Visualization and Computer Graphics **19** (2013), no. 8, 1405–1414.

[AFM⁺06] Jérémie Allard, Jean-Sébastien Franco, Clément Ménier, Edmond Boyer, and Bruno Raffin, *The GrImage Platform : A Mixed Reality Environment for Interactions*, 4th International Conference on Computer Vision Systems, ICVS'06, January, 2006 (New York City, United States), IEEE, 2006, pp. 46–46.

[AHH⁺07] Anick Abourachid, Marc Herbin, Rémi Hackert, Ludovic Maes, and V. Martin, *Experimental study of coordination patterns during unsteady locomotion in mammals*, Journal of Experimental Biology **210** (2007), 366–372.

[BBG⁺10] E.L. Brainerd, D.B. Baier, S.M. Gatesy, T.L. Hedrick, K.A. Metzger, S.L. Gilbert, and J.J. Crisco, *X-ray reconstruction of moving morphology (xromm) : precision, accuracy and applications in comparative biomechanics research.*, Journal of Experimental Zoology PartA **313A** (2010), 262–279.

[Ber10] Alain Berthoz, *La simplexité*, Odile Jacob, Paris, 2010.

[BSF09] Marcus A Brubaker, Leonid Sigal, and David J Fleet, *Estimating contact dynamics*, ICCV, 2009, pp. 2389–2396.

[BSP02] Kiran S. Bhat, Steven M. Seitz, and Jovan Popovic, *Computing the physical parameters of rigid-body motion from video*, Proceedings of the 7th European Conference on Computer Vision-Part I (London, UK, UK), ECCV '02, Springer-Verlag, 2002, pp. 551–565.

[CBI10] Cedric Cagniart, Edmond Boyer, and Slobodan Ilic, *Probabilistic deformable surface tracking from multiple videos*, ECCV, 2010.

[CGMA10] S. Corazza, E. Gambaretto, L. Mundermann, and T.P. Andriacchi, *Automatic generation of a subject-specific model for accurate markerless motion capture and biomechanical applications*, Biomedical Engineering, IEEE Transactions on **57** (2010), no. 4, 806 –812.

[CKJ+11] Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel Van De Panne, *Locomotion Skills for Simulated Quadrupeds*, ACM Transactions on Graphics, Proceedings of SIGGRAPH'11 **30** (2011), no. 4, Article 59.

[CMU02] CMU, *Carnegie mellon university mocap database*, http ://mocap.cs.cmu.edu, 2002.

[DAA+07] S.L. Delp, F.C. Anderson, A.S. Arnold, P. Loan, A. Habib, C.T. John, E. Guendelman, and D.G. Thelen, *Opensim : Open-source software to create and analyze dynamic simulations of movement*, Biomedical Engineering, IEEE Transactions on **54** (2007), no. 11, 1940 –1950.

[DCRB12] Estelle Duveau, Simon Courtemanche, Lionel Reveret, and Edmond Boyer, *Cage-based Motion Recovery using Manifold Learning*, 3DIMPVT 2012 - Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (Zürich, Switzerland), IEEE, October 2012, pp. 206–213.

[DRBR09] Julien Diener, Mathieu Rodriguez, Lionel Baboud, and Lionel Reveret, *Wind projection basis for real-time animation of trees*, Computer Graphics Forum (Proceedings of Eurographics 2009) **28** (2009), no. 2, 289–298, Special Issue : Eurographics 2009.

[FLJ03] P. Thomas Fletcher, Conglin Lu, and Sarang C. Joshi, *Statistics of shape via principal geodesic analysis on lie groups*, 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2003 Proceedings CVPR-03, 2003, pp. –95.

[FRDC04] Laurent Favreau, Lionel Reveret, Christine Depraz, and Marie-Paule Cani, *Animal gaits from video*, ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA), ACM SIGGRAPH / Eurographics, 2004.

[GBT04] Pascal Glardon, Ronan Boulic, and Daniel Thalmann, *Pca-based walking engine using motion capture data*, Proceedings of the Computer Graphics International (Washington, DC, USA), IEEE Computer Society, 2004, pp. 292–298.

[GMHP04] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović, *Style-based inverse kinematics*, ACM Trans. Graph. **23** (2004), 522–531.

[GRP10] Benjamin Gilles, Lionel Reveret, and Dinesh Pai, *Creating and animating subject-specific anatomical models*, Computer Graphics Forum **29** (2010), no. 8, 2340–2351.

[GSdA+09] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel, *Motion capture using joint skeleton tracking and surface estima-*

tion, Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, june 2009, pp. 1746 –1753.

[KCD09]   Marin Kobilarov, Keenan Crane, and Mathieu Desbrun, *Lie group integrators for animation and control of vehicles*, ACM Trans. Graph. **28** (2009), no. 2, 16–1.

[KRFC09] Paul Kry, Lionel Revéret, François Faure, and Marie-Paule Cani, *Modal locomotion : animating virtual characters with natural vibrations*, Computer Graphics Forum (Proceedings of Eurographics 2009) **28** (2009), no. 2, 289–298, Special Issue : Eurographics 2009.

[Lau94]   A. Laurentini, *The visual hull concept for silhouette-based image understanding*, IEEE Trans. Pattern Anal. Mach. Intell. **16** (1994), no. 2, 150–162.

[Law05]   N. D. Lawrence, *Probabilistic non-linear principal component analysis with gaussian process latent variable models*, Machine Learning Research **6** (2005), 1783 – 1816.

[LLCO08] Yaron Lipman, David Levin, and Daniel Cohen-Or, *Green coordinates*, ACM Trans. Graph. **27** (2008), 78 :1–78 :10.

[LMH10]   Martin de Lasa, Igor Mordatch, and Aaron Hertzmann, *Feature-based locomotion controllers*, ACM Transactions on Graphics **29** (2010), Article 131.

[MHKE11] Th.B. Moeslund, A. Hilton, V. Krüger, and L. Sigal (Eds.), *Visual analysis of humans : Looking at people*, Springer, 2011.

[MLS94]   Richard M. Murray, Zexiang Li, and S. Shankar Sastry, *A mathematical introduction to robotic manipulation*, CRC Press, Inc., Boca Raton, FL, USA, 1994.

[MWTK13] Igor Mordatch, Jack M. Wang, Emanuel Todorov, and Vladlen Koltun, *Animating human lower limbs using contact-invariant optimization*, ACM Trans. Graph. **32** (2013), no. 6, 203 :1–203 :8.

[Pen06]   Xavier Pennec, *Intrinsic statistics on riemannian manifolds : Basic tools for geometric measurements*, Journal of Mathematical Imaging and Vision **25** (2006), no. 1, 127.

[RBB00]   Lionel Reveret, Gérard Bailly, and Pierre Badin, *MOTHER : A new generation of talking heads providing a flexible articulatory control for video-realistic speech animation*, Int. Conference of Spoken Language Processing, ICSLP'2000 (Pekin, China), 2000, pp. –.

[RBM+13] Ludovic Righetti, Jonas Buchli, Michael Mistry, Mrinal Kalakrishnan, and Stefan Schaal, *Optimal distribution of contact forces with inverse dynamics control*, The International Journal of Robotics Research **32** (2013), no. 3, 280–298.

[RCM13]   T Robert, J Causse, and G Monnier, *Estimation of external contact loads using an inverse dynamics and optimization approach : General method*

and application to sit-to-stand maneuvers, Journal of biomechanics **46** (2013), no. 13, 2220–2227.

[RE01]   Lionel Reveret and Irfan Essa, *Visual Coding and Tracking of Speech Related Facial Motion*, IEEE CVPR International Workshop on Cues in Communication (Honolulu, United States), 2001.

[RH91]   Marc H. Raibert and Jessica K. Hodgins, *Animation of dynamic legged locomotion*, SIGGRAPH Comput. Graph. **25** (1991), no. 4, 349–358.

[SBB10]   Leonid Sigal, Alexandru O. Balan, and Michael J. Black, *Humaneva : Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion*, Int. J. Comput. Vision **87** (2010), no. 1-2, 4–27.

[TM07]   Lena H. Ting and Lucas J McKay, *Neuromechanics of muscle synergies for posture and movement*, Current opinion in neurobiology **17** (2007), no. 6, 622–628.

[TWC+09]   Maxime Tournier, Xiaomao Wu, Nicolas Courty, Elise Arnaud, and Lionel Reveret, *Motion Compression using Principal Geodesics Analysis*, Computer Graphics Forum, Proceedings of Eurographics'09 **28** (2009), no. 2, 355–364.

[WC10]   Xiaolin Wei and Jinxiang Chai, *Videomocap : modeling physically realistic human motion from monocular video sequences*, ACM Trans. Graph. **29** (2010), no. 4, 42 :1–42 :10.

[WK88]   Andrew Witkin and Michael Kass, *Spacetime constraints*, SIGGRAPH Comput. Graph. **22** (1988), no. 4, 159–168.

[WTR11]   Xiaomao Wu, Maxime Tournier, and Lionel Reveret, *Natural Character Posing from a Large Motion Database*, IEEE Computer Graphics and Applications **31** (2011), no. 3, 69–77.

[YLvdP07]   KangKang Yin, Kevin Loken, and Michiel van de Panne, *Simbicon : simple biped locomotion control*, SIGGRAPH '07 : ACM SIGGRAPH 2007 papers (New York, NY, USA), ACM, 2007, p. 105.

[ZVDH03]   Victor Brian Zordan and Nicholas C. Van Der Horst, *Mapping optical motion capture data to skeletal motion using a physical model*, Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (Aire-la-Ville, Switzerland, Switzerland), SCA '03, Eurographics Association, 2003, pp. 245–250.