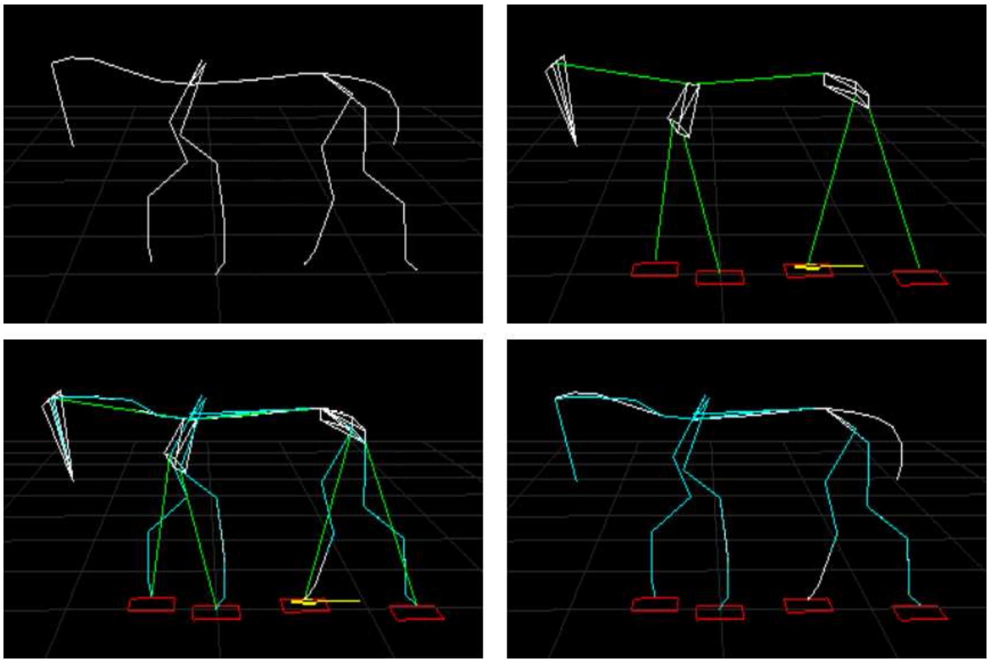
Extension aux quadrupèdes d'un moteur d'animation 3D de personnages

Rapport de Stage - Master II IVR

Nicolas Chaverou



Année 2006

Encadrants : **Lionel Revéret (EVASION - Inria Grenoble) et Franck Multon (LPBEM - Rennes 2)**

Table des matières

Résumé

Le but de ce stage est d'étendre aux animaux quadrupèdes, les caractéristiques d'un moteur d'animation 3D aujourd'hui dédié aux seuls personnages humains bipèdes. Le moteur d'animation existant, MKM (Manageable Kinematics Motions), permet d'adapter automatiquement en temps-réel, un ou plusieurs, mouvements locomoteurs de personnage bipède, à une nouvelle morphologie et à un environnement dynamique.

Il repose sur un paramétrage générique du squelette d'animation, auquel on applique des contraintes cinématiques de positions et de mouvements. Le travail sur l'extension quadrupède reprend les travaux sur le squelette des quadrupèdes de Revéret et al., 2005, pour en proposer un paramétrage universel compatible avec l'approche de MKM. Les résultats attendus seront de pouvoir modifier et mélanger, en temps réel et en fonction du terrain, des animations existantes d'animaux quadrupèdes, disponibles au laboratoire.

Ce stage se fait en collaboration avec l'IRISA et le Laboratoire de Physiologie et de Biomécanique de l'Exercice Musculaire (LPBEM) à Rennes.

Chapitre 1

Introduction

Le Monde de Narnia ou *L'âge de glace* sont, entre autres, des exemples de films, où des quadrupèdes numériques occupent une place centrale. Ainsi, la nécessité d'animer de manière réaliste ces animaux devient de plus en plus importante, et ceci aussi bien dans le domaine du cinéma que dans celui du jeu vidéo.

Si pour les humains, on peut facilement obtenir des mouvements réalistes par le biais de techniques de capture de mouvement, cela s'avère rarement adapté pour les quadrupèdes : elles nécessitent une infrastructure adaptée et des animaux coopératifs. On utilise alors généralement l'animation par *keyframes*. Toutefois, même pour un animateur expérimenté, chaque mouvement représente une tâche longue et complexe. De plus, ce travail est réalisé pour une morphologie spécifique d'animal et pour un environnement prédéfini.
Il pourrait aussi être intéressant d'utiliser des techniques de mélange de mouvements. Ces dernières permettent de créer, à partir d'actions élémentaires, un mouvement plus complexe. Toutefois les techniques existantes s'avèrent peu adaptées aux quadrupèdes évoluant dans un environnement interactif.

Pour répondre à ce manque de flexibilité, nous proposons donc une adaptation pour les quadrupèdes de MKM (Manageable Kinematic Motions [MKM]), un moteur d'animation 3D temps-réel dédié aux bipèdes, developpé par l'IRISA et le Laboratoire de Physiologie et de Biomécanique de l'Exercice Musculaire (LP-BEM) de Rennes. Basé sur la cinématique, il synchronise, mélange et adapte des mouvements automatiquement à des morphologies de squelettes différents. Cette adaptation morphologique se fait par le biais d'une représentation adimensionnelle du squelette d'animation.

Toutefois, la plupart des méthodes et algorithmes utilisés, sont optimisés et uniquement adaptés aux bipèdes. Contrairement à la première intuition que l'on pourrait avoir, un quadrupède est beaucoup plus complexe que la réunion de deux bipèdes. En effet, il existe une certaine dépendance entre ses pattes avant et arrière, qui rend délicate l'adaptation des méthodes de MKM et créée de nouvelles problématiques.

Dans la première partie, après avoir rappelé quelques notions d'animation, nous présenterons le moteur d'animation MKM, à travers l'ensemble de ses fonctionnalités et de ses limites. Ces mêmes limites permettront d'exposer les problématiques à résoudre pour la réalisation de MKM^{quad}, l'adaptation de MKM aux quadrupèdes. Dans une seconde partie, nous détaillerons la démarche adoptée pour résoudre ces problématiques et les solutions retenues. Nous illustrerons, dans une troisième partie, la pertinence de nos choix grâce à une gamme de résultats et tests avant de conclure dans une quatrième partie.

Chapitre 2

État de l'art

2.1 Principes de base de l'animation de personnages

Il existe plusieurs méthodes pour animer un personnage 3D complexe. Les plus répandues d'entre elles, consistent à utiliser un squelette d'animation.

2.1.1 Le squelette d'animation

Que ce soit pour un humain ou un animal, la complexité de son squelette fait que l'on ne peut pas le manipuler dans son ensemble. On décide donc de regrouper certains os et articulations, pour obtenir une version simplifiée qui garde un nombre de degrés de liberté suffisant pour représenter les principaux mouvements. Cette simplification est appelée un *squelette d'animation* (figure 2.1a).

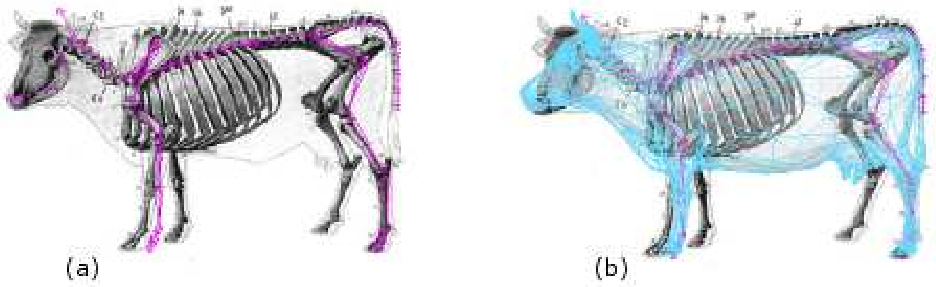


FIG. 2.1 – Un exemple d'un squelette d'animation (en violet) pour une vache.

On peut alors facilement animer ce squelette, et, une fois satisfait de l'animation, on peut lui affecter un maillage 3D complexe (figure 2.1b).

2.1.2 Les méthodes d'animation

Maintenant que toutes les articulations caractéristiques d'un squelette ont été isolées dans un squelette d'animation, il est nécessaire de choisir une méthode pour l'animer. En regardant l'évolution des travaux dans ce domaine, on peut distinguer trois problématiques sur lesquelles se focalisent les recherches :

– rendre *réaliste* les mouvements calculés. L'observateur humain est très sensible aux erreurs qu'un squelette d'animation peut produire. Ces dernières sont difficiles à quantifier et à corriger. Les travaux se focalisent donc à formuler mathématiquement les contraintes qui sont appliquées à un mouvement et qui permettent d'assurer un critère minimal de réalisme.

– rendre les algorithmes moins coûteux en temps de calcul. Les besoins en matière d'animation évoluent toujours : besoin de contrôle interactif, squelettes de plus en plus complexes, animation de foules... Compte tenu de la quantité importante de mouvements réalisables par un humain ou un animal, les recherches se basent surtout sur des familles de mouvements comme la locomotion ou la préhension d'objets.

– rendre le travail de l'animateur plus facile. Les coûts importants d'une production en image de synthèse incitent à développer des méthodes visant à automatiser la production. Les travaux menés vont de l'automatisation de processus liés à l'acquisition à l'édition.

Les nombreuses méthodes visant à résoudre ces problématiques se classent en trois grandes familles :

Les méthodes cinématiques

Introduites par Zelter dans [Zel82], elles cherchent à produire un ensemble d'effets (sans s'intéresser aux causes), à partir d'une description donnée. Ces méthodes ne respectent pas nécessairement les règles de la physique mais se concentrent davantage sur la cinématique liée au squelette (positions, vitesses et accélérations).

La motivation première est de décharger le travail des animateurs qui, au départ, décrivaient le mouvement comme une suite d'images dans le temps. L'idée est de seulement définir des *positions clés* (keyframes), les positions intermédiaires étant ensuite calculées par un type interpolation choisit préalablement. La qualité du mouvement résultant est alors très dépendante du choix des positions clés. Ainsi un échantillonage trop faible ne retranscrira pas les hautes fréquences auxquelles l'oeil humain est sensible, le mouvement aura alors tendance à être trop *lissé*.

Cinématique directe

Elle consiste à spéficier un vecteur d'états à un joint dans le temps. A partir de ce vecteur, et d'une position initiale, on peut alors calculer la position et l'orientation finale de ce joint. Typiquement pour animer un bras en cinématique directe, on va alors spécifier pour chaque articulation, en partant de l'épaule vers la main, un vecteur d'états. Cette méthode n'est pas adaptée pour l'animation de chaînes cinématiques complexes. On voudrait ne pouvoir déplacer que l'extremité d'une chaîne et que les vecteurs d'états des articulations intermédiaires soient calculées automatiquement.

Cinématique inverse

Elle permet de répondre aux limites de la cinématique inverse : à partir d'une position initiale et d'une position finale d'une chaîne cinématique, elle calcule un vecteur d'états pour chaque articulation intermédiaire. Généralement ce calcul se fait par le biais d'une inversion de matrice jacobienne pour chaque articulation. Si un nombre *raisonnable* d'articulations nécessitent des calculs de cinématique inverse avec inversion de matrice jacobienne, les méthodes d'animation cinématique peuvent être intéressantes dans un contexte temps-réel. Il existe des alternatives pour des cas particuliers où la cinématique inverse peut-être calculée de manière analytique : ces dernières seront expliquées un peu plus tard.

Toutefois, comme ces méthodes ne s'intéressent pas aux causes du mouvement, leur principal inconvénient est qu'elles peuvent produire des animations irréalistes.

Les méthodes dynamiques

Introduites pour la première fois par Isaacs et al. dans [IC87], elles sont basées sur les lois fondamentales de la dynamique qui s'intéressent plus particulièrement aux causes du mouvement (représentées par des forces) et aux accélérations, en tenant compte des masses corporelles et de leur inertie.

Les méthodes cinématiques, vues précédemment, permettent de modifier les mouvements d'origine à l'aide de contraintes liées à sa nature : sans connaissance spécifique sur ces derniers, les déformations peuvent produire des actions irréalistes. Les méthodes dynamiques permettent de pallier ce problème en utilisant des lois universelles de la physique (formule de Newton-Euler, formule de Lagrange). Les mouvements produits sont donc très proches de la réalité.

Dans [LP02], Liu ., présentent une méthode qui permet, à partir de quelques positions clés (figure 2.2a et b) et des ordres tels que *marche*, *cours* ou *saute*, d'interpoler dynamiquement un mouvement (figure 2.2c). L'animation résultante est très réaliste mais nécessite plusieurs heures de calcul pour trouver l'ensemble des forces à appliquer et n'est donc pas adaptée aux applications temps-réel.

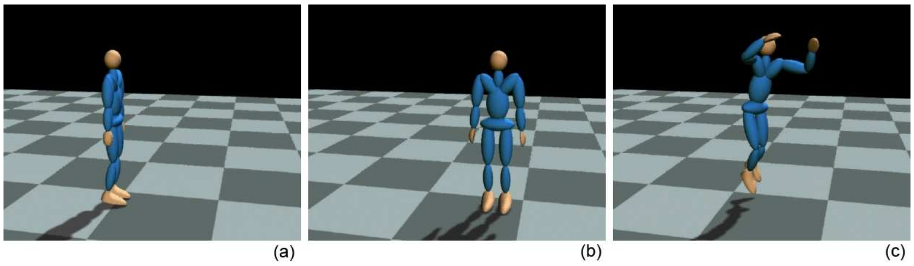


FIG. 2.2 – Interpolation dynamique entre des positions clés.

Certaines méthodes, comme celle de Laszlo et al. dans [LvdPF00], ont tenté une approche temps-réel de l'animation dynamique. L'idée est de fournir à l'utilisateur un ensemble de contrôles, via le clavier ou la souris, permettant d'animer différentes parties d'un squelette quelconque. On peut alors déplacer à son gré n'importe quel personnage en respectant les lois de la physique. Le principal inconvénient de cette méthode est que l'utilisateur doit contrôler simultanément les différentes parties du squelette : on compte 12 touches pour contrôler seulement deux pattes d'un chat en 2D (figure 2.3), le passage en 3D nécessiterait alors un trop grand nombre de contrôles, ingérables pour un seul utilisateur.

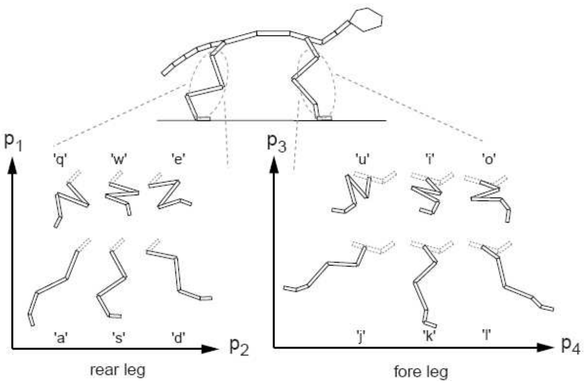


FIG. 2.3 – Contrôles interactifs des pattes d'un chat en 2D.

Au contraire des méthodes cinématiques, les méthodes dynamiques génèrent à coup sûr des animations valides et minimisent l'intervention d'un animateur. La contrepartie est que ces animations nécessitent de longues heures de calcul, dûes au fait qu'elles cherchent l'ensemble des mouvements mécaniquement réalisables. De plus, la plupart du temps le contrôle moteur en choisit une qu'on a du mal à caractériser encore à l'heure actuelle. Enfin, les quelques approches temps-réel restent encore inexploitables.

Les méthodes cinétiques

Ces méthodes sont une extension de la cinématique dans laquelle on sait gérer les masses et, donc, le centre de masse. Dans des conditions quasi-statiques (accélération nulle), on respecte alors la dynamique. Le calcul des positions de centres de masse est donc une alternative pour résoudre les inconvénients des méthodes cinématiques. Ces positions sont calculées analytiquement et permettent, si besoin, de corriger une posture grâce à quelques de calculs de cinématique inverse.

Le résultat n'est pas nécessairement dynamiquement valide selon le choix et le nombre des centres de masse calculés, mais il aura, au moins, l'air réaliste pour un observateur humain.

2.2 MKM : Analyse de l'existant

Issue de la collaboration de l'équipe SIAMES de l'IRISA et du laboratoire LPBEM de l'Université de Rennes 2, MKM (Manageable Kinematics Motions) est une librairie permettant d'animer en temps-réel des humanoïdes synthétiques. Elle se décompose en deux parties (figure 2.4) :

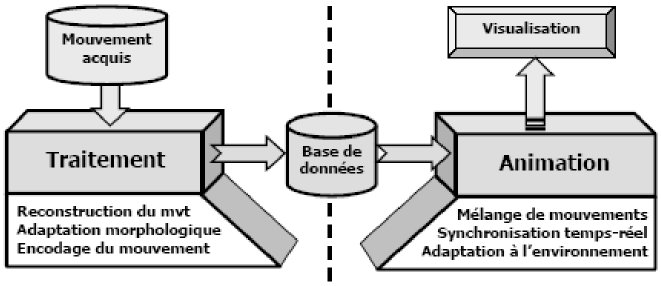


FIG. 2.4 – Représentation générale de MKM.

– une partie traitement : elle permet d'éditer le mouvement acquis et d'y ajouter manuellement des contraintes principales (contraintes d'appui) et secondaires (contrainte de distance, position...). Ces contraintes permettent d'adapter l'action à des squelettes de morphologie différente. Une fois les contraintes spécifiées, le mouvement est encodé à l'aide d'un squelette normalisé. On ne garde ainsi que les informations caractéristiques du mouvement, indépendantes du modèle morphologique de base.

– une partie animation : elle permet de mélanger les mouvements pré-traités et de les affecter à n'importe quelle morphologie d'humanoïdes. Elle permet ensuite d'adapter cet humanoïde en mouvement à l'environnement (qui peut-être interactif). Contrairement à la partie traitement, la partie animation se fait en temps-réel.

Afin de bien cerner l'ensemble des problématiques résultantes de l'adaptation du moteur bipède aux quadrupèdes, il est nécessaire d'en étudier les composantes.

2.2.1 Les fonctionnalités de MKM : La partie traitement

Avant d'affecter des mouvements à un personnage, il est nécessaire de traiter ces mouvements afin qu'ils puissent être utilisables dans n'importe quel environnement et pour n'importe quelle morphologie. Dans un premier temps, les mouvements sont encodés, via une représentation spéciale du squelette d'animation. Cependant, un mouvement ne se limite pas qu'à une séquence de postures, mais aussi à des contraintes. Un mouvement est donc stocké en utilisant une représentation normalisé de l'action et un ensemble de contraintes associées.

Le squelette adimensionnel

C'est dans cette représentation du squelette d'animation que réside toute la force de MKM (figure 2.5). En n'utilisant que des données normalisées, on s'abstrait de toute dépendance anthropologique et de nombreux calculs sont ainsi simplifiés. En effet, les représentations classiques de bipède nécessitent l'utilisation de cinématique inverse avec inversion de matrice jacobienne pour résoudre certaines contraintes telles que se déplacer dans un environnement interactif ou s'assurer que les pieds sont en contact avec le sol.

Pour résoudre ces contraintes sans utiliser de lourdes méthodes lourdes de cinématique inverse (avec inversion de matrice jacobienne pour chaque degré de liberté), Kulpa et al. dans [KMA05], proposent de diviser le squelette en sous parties exprimées relativement à leurs parents et auxquelles on associe des coordonnés cartésiennes et des données angulaires. En effet, plutôt que de stocker les angles de chaque joint, seuls sont stockés les angles indépendants de la morphologie (comme l'angle du plan contenant l'épaule, le coude et le poignet). Les données cartésiennes sont normalisées de manière à être adimensionnelles et à simplifier le processus de mise à l'échelle (*scaling*)

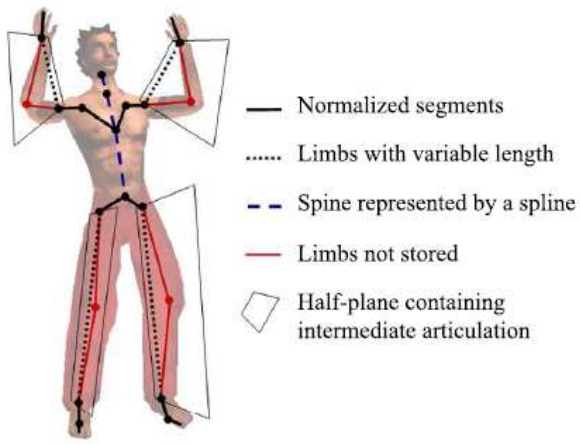


FIG. 2.5 – Représentation normalisée d'un squelette de bipède.

Classiquement, le corps humain est considéré comme un ensemble de chaînes cinématiques partant de la racine aux extrémités (tel que les bras, les jambes...). Dans cette représentation, ces chaînes sont divisées en trois grands types :

- les segments normalisés de longueur fixe qui sont composés d'un seul segment de corps (mains, pieds...).
- les segments normalisés de longueur variable qui sont composés de deux segments de corps (bras, jambes). Les joints intermédiaires (genoux et coudes) se sont pas stockés car leurs positions sont directement liées à la morphologie du personnage. Une méthode analytique, décrite plus tard, permet de retrouver leurs

positions grâce à une méthode analytique et rapide de cinématique inverse introduite par Tolani et al. dans [TGB00].
– une spline qui représente la colonne vertébrale et qui peut être divisée en autant de segments voulus dans la partie animation temps-réel.

Quelque soit le type de la chaîne composées de segments S_j, seules des valeurs normalisées sont stockées. Soit E et R, respectivement l'extrémité et la racine d'une chaîne KC. La représentation normalisée de KC est obtenue par l'équation suivante :

$$normalizedKC = \frac{E - R}{\sum_j length(S_j)}$$

.
Pour chaque *frame*, les bras et les jambes sont stockés en utilisant deux informations : un plan de référence attaché à la racine de la chaîne cinématique (épaule ou hanche) et un scalaire. Le scalaire correspond à la longueur normalisée du membre en question ; le pourcentage de la longueur maximum de celui-ci (voir formule de calcul ci-dessus).

Le plan correspond au domaine de validité de l'articulation intermédiaire : l'ensemble des positions dans lesquelles elle peut se trouver (figure 2.6). En effet, que ce soit pour les humains ou pour les animaux, les rotations sur les articulations intermédiaires, ne se font pas sur un axe fixe et ont un peu de jeu au niveau de cet axe. Toutefois par mesure de simplification, en animation, il est admis qu'ils n'ont qu'un degré de liberté.

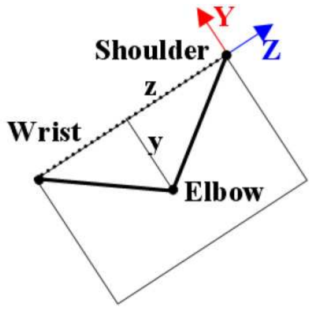


FIG. 2.6 – Spécification du demi-plan représenté par le cadre (X, Y, Z) qui contient le coude. Les coordonnées du coude (y, z) sont calculées analytiquement.

Une fois le mouvement encodé, on peut lui associer un ensemble de contraintes.

Les contraintes

On distingue deux types de contraintes : les contraintes principales (contraintes d'appui) et secondaires (contrainte de distance, position...). Pour l'instant, les contraintes secondaires ne sont pas encore disponibles dans le moteur, les informations sur celles-ci sont donc juste fournies à titre indicatif.

Les contraintes principales

Dans un moteur d'animation, il est primordial de savoir quand un pied est en contact avec le sol : ce contact est appelé une phase de support. Ces phases de support sont ensuite exploitées par le module de synchronisation de la partie animation pour vérifier si deux mouvements peuvent être mélangés.

Le calcul de ces contraintes se fait semi-automatiquement grâce au réglage de deux paramètres (figure 2.7) :

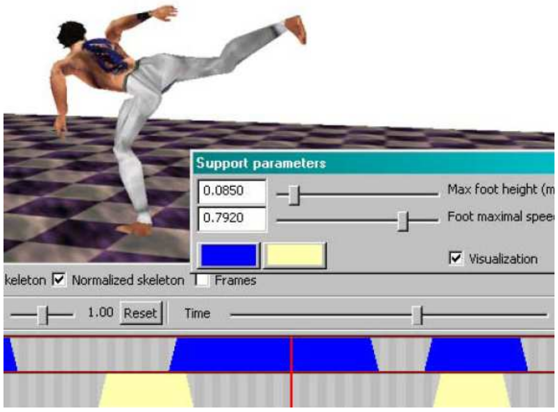


FIG. 2.7 – Interface du générateur de phases de support. En bas, les deux zones horizontales définissent les phases de support pour chaque pied

– la hauteur maximale autorisée pour l'extrémité la plus basse du pied. Cela représente la hauteur au delà de laquelle le pied n'est pas en contact avec le sol
– la vitesse maximale autorisée quand le pied est en dessous du seuil précédent. Cela permet de discriminer les actions qui frôlent le sol à une forte vitesse et qui ne sont pas une phase de support (coup de pied circulaire par exemple).

Il peut être également utile de définir d'autres types de contraintes.

Les contraintes secondaires

Ces contraintes sont associées à un segment du squelette et sont gérées par un moteur temps-réel de cinématique et cinétique inverse. Ainsi une contrainte sur la main peut nécessiter le tronc si c'est nécessaire. De plus, l'utilisation de la cinétique permet d'utiliser automatiquement les membres disponibles afin d'assurer l'équilibre du personnage. On distingue plusieurs types de contraintes secondaires : préhension, position, distance, espace de liberté...

Le mouvement encodé et l'ensemble de contraintes associées sont stockées dans un fichier S4D, propre à la librairie et exploitable par la partie animation.

2.2.2 Les fonctionnalités de MKM : La partie animation

Elle est composée de différents modules qui opèrent en temps-réel (figure 2.8).

Une fois les mouvements traités, l'utilisateur peut, à l'aide de priorités, en affecter, un ou plusieurs, à un personnage. Afin de pouvoir être mélangées pertinemment, ces actions sont tout d'abord synchronisées entre elles (*module Synchronization*). À partir de cette synchronisation, on peut extraire la posture d'un mouvement au temps t (*module Posture*). Ces postures sont ensuite *scalées* sur le squelette d'un personnage pour prendre en compte une partie de sa morphologie (*module Morphological adaptation*), puis sont mélangées en tenant compte des priorités(*module Blending*). Pour obtenir la posture finale, il est encore nécessaire d'adapter le personnage à son environnement et enfin de calculer les positions de ses articulations intermédiaires (*module Environmental adaptation*).

Le mélange de mouvements est une technique qui permet de produire une action complexe à partir de petites actions simples. Malheureusement les algorithmes de mélange peuvent produire des résultats invalides : si

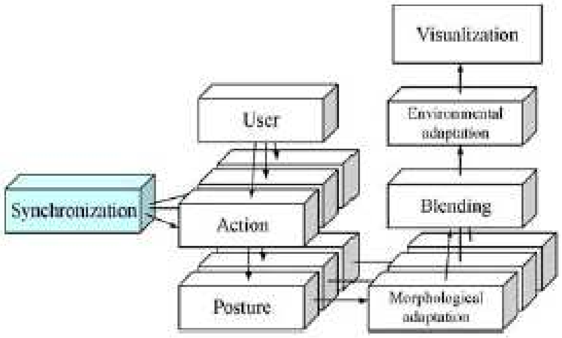


FIG. 2.8 – Représentation globale des modules de MKM.

un mouvement où le pied gauche au sol est mélangé avec un mouvement où le pied droit est au sol, le résultat ne respectera aucune des contraintes de support des mouvements d'origine. Cela entraînera un glissement des pieds sur le sol. Pour éviter ces problèmes, les mouvements doivent être précédemment synchronisés.

Le module de synchronisation

Il s'agit de la première étape de la partie animation. Introduite par Stéphane Ménardais dans sa thèse [Mén03] puis précisée dans [MKMA04], cette méthode permet de synchroniser plusieurs mouvements en temps-réel. Les séquences de points d'appuis des pieds sont très importantes dans un moteur d'animation de bipèdes. On peut décomposer une action A_i en une suite de phases de support S_i (figure 2.9). Ces phases ont été définies dans la partie traitement, lors de l'ajout de contraintes d'appui. On peut les représenter par un ensemble à cinq éléments $FS = \{RS, LS, DS, \emptyset, Err\}$:

- *RS* définit la contrainte d'appui unipodal droit.
- *LS* définit la contrainte d'appui unipodal gauche.
- *DS* définit la contrainte d'appui bipodal.
- $\emptyset$ définit la contrainte sur laquelle aucun appui n'est présent (un saut par exemple).
- *Err* définit un élément caractérisant une contrainte impossible.

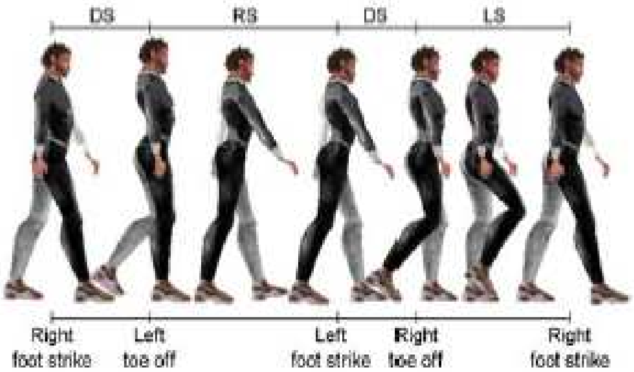


FIG. 2.9 – Un exemple des phases de support pour une marche

Une nouvelle relation algébrique

Afin de ne pas mélanger des mouvements incompatibles (cloche-pied à gauche et cloche-pied à droite par exemple), il est nécessaire de les synchroniser. Pour cela, on introduit la relation algébrique $\oplus$. Cette relation de $FS^2 \rightarrow FS$, est définie de telle manière que a$\oplus$b, représente la séquence de support résultante du mélange des séquences a et b (figure 2.10).

$\oplus$	DS	RS	LS	Err	Ø
DS	DS	RS	LS	Err	Ø
RS	RS	RS	Err	Err	Ø
LS	LS	Err	LS	Err	Ø
Err	Err	Err	Err	Err	Ø
Ø	Ø	Ø	Ø	Ø	Ø

FIG. 2.10 – Allure de la relation $\oplus$ qui détermine si deux mouvements peuvent être mélangés ou non.

Soit $S_i(k)$ le k-ième élément de l'action A_i. Comme cette relation est commutative, deux actions A_i et A_j sont synchronisées sur *nk* phases de support si et seulement si :

$$\forall k, S_i(k) \oplus S_j(k) \neq Err$$

Et de manière plus générale, comme la relation est associative, *n* actions sont synchronisées sur *nk* phases de support si et seulement si :

$$\forall k, \oplus_{i \in [1,n]} S_i(k) \neq Err$$

L'algorithme de synchronisation

Avant de mélanger deux actions, le module va tenter de synchroniser *nk* phases de support (cad ne provoquant pas d'erreurs). La relation algébrique introduite précedemment, permet alors de décrire l'algorithme de synchronisation. Celui ci se divise en 3 étapes :

1. l'organisation des actions à synchroniser. Due à l'interactivité du moteur, un utilisateur peut démarrer un mouvement quand il le veut. L'algorithme doit alors pouvoir synchroniser *nk* phases. Cela implique que le mouvement joué doit être assez long pour être synchronisé. Dans le cas d'un mouvement cyclique, on décide de répéter celui-ci. Dans le cas des autres mouvements, on répète la dernière phase de support (figure 2.11).

A_i	$S_i(1)$	$S_i(2)$	$S_i(3)$	$S_i(4)$	$S_i(5)$	$S_i(6)$	$S_i(7)$	$S_i(8)$
A_1	RS	DS	LS	DS	RS	DS	LS	DS
A_2	LS	DS	Ø	DS	LS	LS	LS	LS

Original motion Chronological order

FIG. 2.11 – Exemple d'extension d'un mouvement cyclique et d'un mouvement non-cyclique.

2. la modification des phases de support. Il est fort possible que deux phases de support ne soient pas synchronisées (cad provoquent une erreur), il est donc nécessaire de modifier un des mouvements

pour remédier à cela. Supposons que plusieurs mouvements sont synchronisés à l'étape *k+1* mais pas à l'étape *k+2*. Il suffit alors d'étendre en *k+2*, une des dernières phases de support synchronisée en *k+1* (figure 2.12). Un théorème associé à la relation algébrique et détaillé dans [Mén03] montre qu'il existe toujours au moins une solution. Dans le cas de plusieurs solutions, on va chercher celle qui permet de respecter le plus fidélement le mouvement qui a la plus forte priorité.

Plusieurs actions peuvent donc maintenant être synchronisées sur *nk* phases de support après la phase active. Toutefois, cela nécessite que la phase active soit elle aussi synchronisée. Quand l'utilisateur lance une action, on cherche alors à synchroniser son premier élément avec les éléments courants : le départ d'une action est donc reporté tant que cette condition n'est pas respectée.

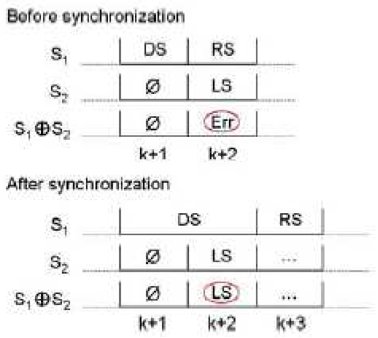


FIG. 2.12 – Exemple d'extension d'une phase de support pour valider le mélange de deux mouvements.

3. la rectification d'éventuelles discontinuités de vitesse. Modifier la durée d'une phase de support, comme expliqué dans l'étape précédente, peut provoquer des discontinuités de vitesse gênantes et ne respectant pas les lois de dynamique. Soit *v(t)* la dérivée de la courbe de vitesse de la phase de support *k* au temps courant *t* (figure 2.13a). Si on étend cette phase (figure 2.13b), il faut modifier sa courbe de vitesse. Pour cela on recalcule la dérivée à l'aide du symétrique borné de *v(t)* (figure 2.13c et d) et on calcule une cardinal spline (figure 2.13f) qui recouvre *k*.

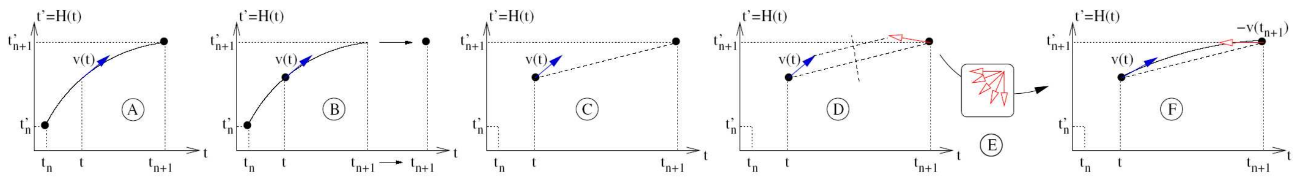


FIG. 2.13 – Exemple de reconstruction d'une courbe de vitesse d'une phase de support étendue.

La constante *nk* permet de préciser sur combien de phases de support, les mouvements doivent être mélangés. Cette constante peut-être discutée : si elle est trop petite, il se peut qu'on ait une déformation temporelle pour passer d'une action à une autre. Si elle est trop grande, l'algorithme devra synchroniser un grand nombre de phases avant de lancer le mélange.

Le module de mélange de mouvement

Comme nous venons de le voir, le module de synchronisation étend les phases de support d'un mouvement si cela est nécessaire. A partir de cette synchronisation, on peut alors, pour chaque mouvement, extraire

une posture normalisée de squelette au temps *t*.
Cette posture correspond donc à un ensemble d'angles, de scalaires et de contraintes à respecter à un moment donné d'un mouvement. Tous ces paramètres, dénotés $Param_i(t)$ (où i correspond à une partie du corps ou une contrainte), définissent une posture au temps t qui sont regroupés dans une structure appelée $Param(t) = \{Param_i(t)\}$.

Détaillé par Ménardais et al. dans [MMKA04], le rôle de ce module est de mélanger, pour chaque mouvement, les postures extraites pour chaque temps *t*, en tenant compte des priorités renseignées par l'utilisateur.

Chaque action A^j est donc associée à une priorité P_i^j pour chaque partie du corps ou contrainte *i*. Par exemple, on définira une priorité haute pour les jambes et une basse pour le haut du corps dans un mouvement de locomotion. Ces priorités peuvent changer en temps-réel selon les besoins de l'utilisateur.

Considérons deux actions affectées à un personnage : l'une de locomotion, l'autre de préhension. L'action de préhension est programmée de telle sorte que la main doit atteindre une certaine position ; mais la main est aussi influencée par l'action de locomotion. Pour résoudre ce problème, un seuil constant a été définit : $\triangle P$. Si on a deux actions A^1 et A^2 de priorités P^1 et P^2 telles que $P^2 > P^1 + \triangle P$, alors on dit que l'action A^1 est négligeable et elle ne sera donc pas prise en compte lors du mélange.

Soit $\Phi(P^1, P^2)$ la fonction qui retourne un réel entre 0 et 1, qui informe sur l'importance de P^1 par rapport à P^2. Si $P^1 > P^2$ alors $\Phi(P^1, P^2) = 1$ (P^1 est plus importante que P^2). Si $P^1 < P^2 - \triangle P$ alors $\Phi(P^1, P^2) = 0$ (P^1 est négligeable par rapport à P^2). Pour les cas intermédiaires tels que $P^2 - \triangle P < P^1 < P^2$ (P^1 est moins importante que P^2 mais pas négligeable), le résultat de $\Phi(P^1, P^2)$ est donné par l'équation suivante :

$$\Phi(P^1, P^2) = \exp\left(\ln(\varepsilon) \times \left(\frac{P^2 - P^1}{\triangle P}\right)^2\right)$$

A partir de cette équation, on peut alors calculer pour chaque action l'ensemble I^j des actions non négligeables par rapport à A^j.

On peut alors définir une fonction $F(j, i)$ qui retourne $Param_i(t)$, la posture mélangée. Cette fonction récursive prend en entrée l'action A^n ayant la plus haute priorité : on s'assure ainsi qu'elle aura une influence sur le mouvement résultant. On calcule ensuite les effets des autres actions j (dans l'ordre inverse des priorités) jusqu'à ce qu'on tombe sur une action négligeable ou jusqu'à ce que toutes les actions aient été traitées. On obtient alors la formule suivante :

$$F(j, i) = G(j, i) + F(j - 1, i)$$

avec

$$G(j, i) = \frac{\sum_{p \in I^j} \left(\Phi\left(P^p, P^j\right)\right) \times Param_i}{\sum_{p \in I^j} \left(\Phi\left(P^p, P^j\right)\right)}$$

Grâce à ces fonctions, un utilisateur peut alors de manière interactive affecter une ou plusieurs actions, et faire varier leurs priorités pour des actions de préhension par exemple (figure 2.14).
Maintenant que les postures normalisées sont correctement mélangées dans une nouvelle posture, il est nécessaire d'affecter celle-ci à une morphologie de personnage. Cette étape se fait rapidement car la posture normalisée ne contient que des angles et des scalaires. Pour rappel, le scalaire correspond à la longueur normalisée du membre en question ; le pourcentage de la longueur maximum de celui-ci. A partir des données morphologiques d'un personnage, on peut alors *scaler* la posture en calculant la nouvelle longueur des différents membres.

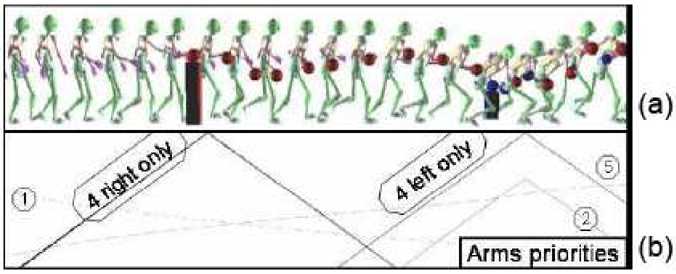


FIG. 2.14 – Mélange de mouvements : (a) séquence résultante du mélange d'une action de marche et de deux actions de préhension, (b) évolution des priorités des bras.

Le module d'adaptation à l'environnement

Détaillé par Kulpa et al. dans [KMA05], ce dernier module permet d'adapter un squelette *scalé* à l'environnement et de calculer les positions des articulations intermédiaires.

Adaptation au sol

L'adaptation au sol consiste à adapter le squelette *scalé* à partir des empreintes (*footprints*) placées sur le sol. Ces empreintes sont calculés à partir des contraintes d'appui stockées dans le fichier de mouvement. Une fonction du moteur permet de récupérer la hauteur du sol (axe Z) pour une position 2D (X, Y) : les nouvelles empreintes sont alors placées automatiquement.

Le module s'assure que la racine du squelette respecte deux conditions : la première est qu'elle doit être le plus proche possible de la hauteur originale $h1$ afin de préserver le style original. La seconde est que la racine doit pouvoir être atteinte à partir de la positions des empreintes et de la longueur maximale des jambes (donnée par $h2$ et $h3$). La hauteur du root est alors le minimum entre ces trois hauteurs : $hRoot = min(h1, h2, h3)$ (figure 2.15).

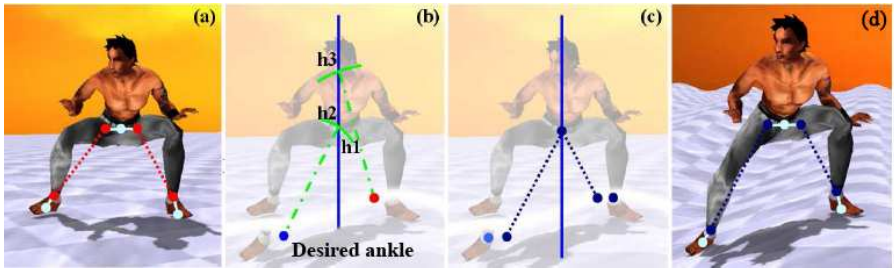


FIG. 2.15 – Adaptation au sol : (a) posture originale, (b) posture à atteindre et hauteurs considérées, (c) la hauteur la plus basse est choisie, (d) posture finale, les jambes sont reconstruites automatiquement.

Calcul des articulations intermédiaires

Une fois les calculs d'adaptation du squelette terminés, on peut déterminer la position des articulations intermédiaires. L'avantage de ne les calculer qu'à la fin, permet ainsi de ne pas les prendre en compte lors des diverses adaptations, on économise alors des calculs coûteux de cinématique inverse.

L'articulation du coude est placée à l'intersection de deux sphères (cad un cercle). La première est centrée sur l'épaule et son rayon est égal à la taille de l'os reliant l'épaule au coude (humérus). La seconde est centrée sur le poignet et son rayon est égal à la taille de l'os reliant le coude au poignet (cubitus). Le coude est alors placé à l'intersection du cercle et du plan de référence, dont l'orientation est stockée dans le fichier S4D. De plus, comme le plan est définit dans le repère de l'épaule, la position relative du coude est facilement calculable :

$$z = \frac{humerus^2 - cubitus^2 + limb^2}{2 \times limb}$$

$$y = \sqrt{humerus^2 - z^2}$$

où z et y sont les coordonnées locales du coude dans le plan de l'épaule (figure 2.6), $humerus$ et $cubitus$ respectivement la longueur de l'humérus et du cubitus, et $limb$ la longueur courante entre l'épaule et le poignet. Le genou est calculé de la même manière pour la jambe.

Respect de la dynamique

Le principal objectif de MKM est de produire des animations temps-réel. Comme, par définition, les mouvements acquis sont dynamiquement valides, ce sont les méthodes d'animation cinématique qui ont été retenues. De plus, les représentations choisies, expliquées plus tôt, permettent d'éviter les coûteux calculs de cinématique inverse pour retrouver les articulations intermédiaires. Comme beaucoup d'autres systèmes basés sur la cinématique, certaines limitations apparaissent. En effet, rien ne garantit que le mélange de deux actions dynamiquement valides, le sera lui aussi.

Pour pallier à cela, une méthode cinétique, décrite par Multon et al. dans [KM05], est utilisée ponctuellement dans MKM : pour certaines actions, un ou plusieurs centres de masse sont calculés et sont utilisés pour corriger la posture grâce à quelques calculs rapides de cinématique inverse. Ainsi, chaque chaîne cinématique (bras, jambe...) dispose d'une méthode analytique pour déplacer son centre de masse au bon endroit.

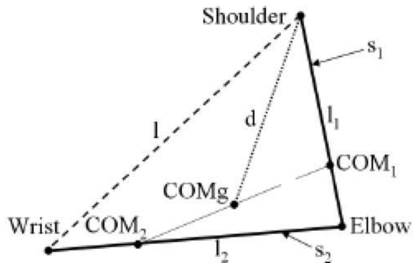


FIG. 2.16 – Calcul du centre de masse d'un bras : COM_1 (et resp. COM_2) est la position du centre de masse du segment s_1 (et resp. s_2), COM_g est la position du centre de masse du membre.

Le centre de masse d'un bras est ainsi obtenu à partir des centres de masse des deux segments qui le composent (figure 2.16). Les positions de ces derniers sont calculés depuis l'articulation proximale (épaule pour l'humérus et coude pour l'avant bras) en utilisant deux pourcentages $r1$ et $r2$ de la taille du segment. Enfin, le centre de masse du membre est calculé en utilisation un troisième ratio $r3$. Ces ratios sont donnés par les tables anthropométriques trouvées par Zatsiorski dans [ZSC90].

Des calculs de cinématiques inverses sont alors ensuite utilisé pour faire correspondre le centre de masse actuel au centre de masse calculé, tout en respectant l'ensemble des contraintes fixées. Si cela n'est pas possible, les contraintes sont prioritaires sur le centre de masse.

2.2.3 Ses limites

Grâce à une représentation normalisée du squelette bipède et une nouvelle relation algébrique, MKM permet de mélanger plusieurs mouvements de manière pertinente et de les affecter à différentes morphologies de personnages. Toutefois ses méthodes sont optimisées et très spécialisées et donc, pour la plupart, inexploitables pour des quadrupèdes. De plus, contrairement aux intuitions que l'on pourrait avoir, un quadrupède n'est en aucun cas la combinaison de deux bipèdes : en effet, il existe une dépendance entre ses pattes antérieures et ses pattes postérieures démontrée par Cartmill et al. dans [CLS02]. Les cycles de locomotion ou de sauts sont d'ailleurs très liés à cette dépendance, et son non-respect génèrerait alors des animations invalides.

Squelette générique

Malgré le fait que le squelette anatomique d'un humain possède les mêmes caractéristiques que celui d'un quadrupède, le modèle de squelette proposé par MKM n'est pas suffisant pour représenter de manière assez précise les mouvements d'animaux. En effet, se déplaçant sur 4 pattes, ceux-ci possèdent un modèle de locomotion (plantigrade, digitigrade, onguligrade) qui fait travailler beaucoup plus d'articulations que pour les humains. La présence d'une seule articulation intermédiaire (coude, genou) n'est donc pas suffisante pour représenter de manière distincte et fidèle ces différents modèles.

Synchronisation

Le module de synchronisation de MKM se base sur une nouvelle relation algébrique et une matrice associée. Cette matrice donne la phase de support résultante du mélange de deux phases de support. Pour un quadrupède, on distingue 16 types différents de phases de support. On aurait alors une matrice 16x16 et plus de 40% de cas provoquant une erreur à l'intérieur au lieu d'un seul pour les bipèdes. Le théorème de Ménardais dans [Mén03], garantissant qu'il existe toujours au moins une solution si on prolonge une phase de support précedemment synchronisée, ne pourrait plus être vérifié.

Utiliser deux matrices (pour les pattes avant et pour les pattes arrière) ne garantit pas non plus le respect de la dépendance entre les pattes. On pourrait ainsi se retrouver avec les pattes arrières en configuration de saut (aucune patte au sol) provenant d'une prolongation d'une phase de support de saut, pendant que les pattes avant continuerait à effectuer un cycle de marche.

La relation algébrique utilisée dans MKM pour les bipèdes n'est donc pas réutilisable dans une version quadrupède.

Centres de masse

Issus d'observations et de reproductions de vidéos, les mouvements de MKM^{quad} sont par définition dynamiquement valides mais rien ne garantit que le mélange de plusieurs mouvements ou leur adaptation au sol le sera aussi. Dans [ZSC90], Zatsiorsky et al. ont mis au point une table de ratios pour faciliter le calcul des centres de masse pour les humains.
Aucun équivalent de cette table n'existe pour les quadrupèdes. De plus, il en faudrait une par espèce.

Pour résoudre tous ces problèmes, deux alternatives sont possibles : la première est de conserver les principes des méthodes de MKM et proposer des modifications pour les points problématiques. La seconde consiste à complètement changer d'approche pour proposer une nouvelle méthode efficace et adaptée aux quadrupèdes. Dans tous les cas, l'alternative choisie doit permettre de respecter les contraintes du moteur, à savoir permettre d'animer plusieurs personnages de différentes morphologies dans un environnement interactif en temps-réel .

Pour cela, il est nécessaire d'étudier les précédentes recherches menées dans les domaines des méthodes problématiques de MKM.

2.3 Synchronisation et quadrupèdes

Comme nous l'avons vu, toutes les méthodes de MKM ne sont pas adaptées aux quadrupèdes. Il est donc nécessaire de réadapter certaines de ces méthodes, voire de changer d'approche.

Contrairement aux animaux, de nombreux travaux de recherches ont été menés sur l'animation d'humains et sur la synchronisation de mouvements.

2.3.1 Synchronisation de mouvements

On peut différencier deux approches en matière de synchronisation de mouvements. La première consiste à détecter les correspondances entre les mouvements. Ces correspondances sont ensuite utilisées pour trouver une synchronisation unique. La seconde consiste à trouver des transitions entre les postures de différents mouvements. Les transitions sont alors un ensemble de solutions possibles : une fonction d'erreur est généralement utilisée pour trouver la solution la plus appropriée.

Correspondances entre les mouvements

Les correspondances entre les mouvements sont principalement identifiées en utilisant les informations fréquentielles des mouvements. Dans [BW95], Bruderlin et al,. partent de l'intuition que les basses fréquences d'un mouvement correspondent aux traits principaux alors que les hautes fréquences correspondent aux détails et aux subtilités du mouvement. A l'aide d'un filtre *lowpass*, ils isolent alors les basses fréquences de chaque articulation pour chaque mouvement (figure 2.17a) et vont les interpoler pour transiter d'un mouvement à un autre (figure 2.17a).

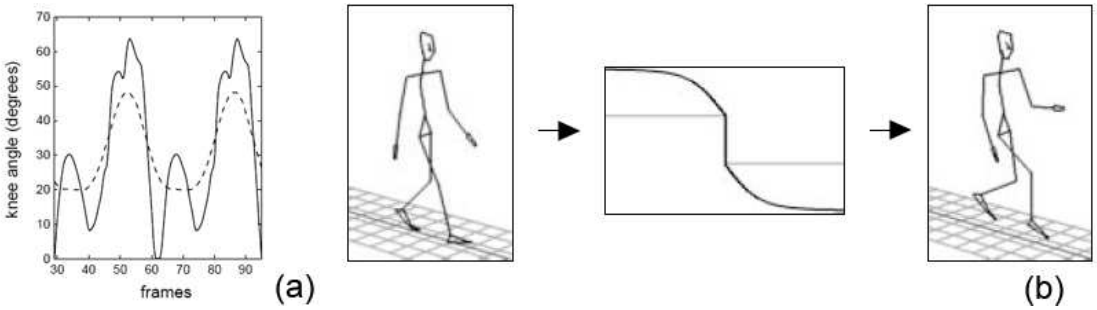


FIG. 2.17 – Méthode de Bruderlin et al. : (a) isolation des basses fréquences du mouvement du genou, (b) interpolation de fréquences entre deux mouvements

Cette méthode, en plus de marcher en temps-réel, a des résultats plutôt convaincants. De plus, elle est facilement extensibles aux quadrupèdes puisqu'elle ne se base que sur les fréquences des mouvements. Toutefois, son principal inconvénient est que les mouvements à synchroniser doivent être similaires dynamiquement parlant, sans quoi les interpolations pourraient être invalides.

Dans [GRG94], Guo et al. utilise une structure appelée *frame space* pour passer d'une allure cyclique à une autre. Sur la figure 2.18, on peut voir un *frame space* à 3 dimensions. Les axes de cet espace représentent la hauteur maximum du pied dans le mouvement (H), la longueur du pas (L) en fonction du temps (T). Les

bornes de cet espace correspondent alors à 4 mouvements locomoteurs (marche et course rapide, marche et course lente).
Au milieu de cet espace, l'utilisateur peut définir un chemin pour passer d'une allure à une ou plusieurs autres, et spécifier une vitesse sur ce chemin. La transition est alors calculée par une interpolation entre les axes du repère.

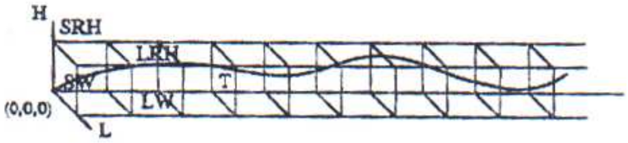


FIG. 2.18 – Exemple d'un *frame space* à 3 dimensions dans lequel est définit un mouvement

Ici aussi la méthode génère des résultats convenables mais elle oblige les mouvements à être cycliques. De plus, ces derniers similaires dynamiquement parlant afin de pouvoir trouver des paramètres pour construire le *frame space*. Or, comme nous voulons synchroniser des actions très différentes (marche, sauts), ces méthodes ne sont pas adaptées.

Transitions entre les postures de différents mouvements

Des recherches plus récentes ont mis au point des algorithmes pour trouver, dans une base de données, des transitions entre plusieurs mouvements. A partir de cette base de données, Kovar et al. dans [KGP02], compilent une structure appelée *Motion graph* qui encode la manière dont les différents mouvements peuvent être mélangés. Le *Motion graph* est un graphe orienté où les arêtes correspondent à des morceaux de mouvements originaux ou à des mouvements de transition générés automatiquement. Les noeuds, eux, servent de *carrefours* entre les différents morceaux de mouvements qui peuvent se concaténer *sans accros*.

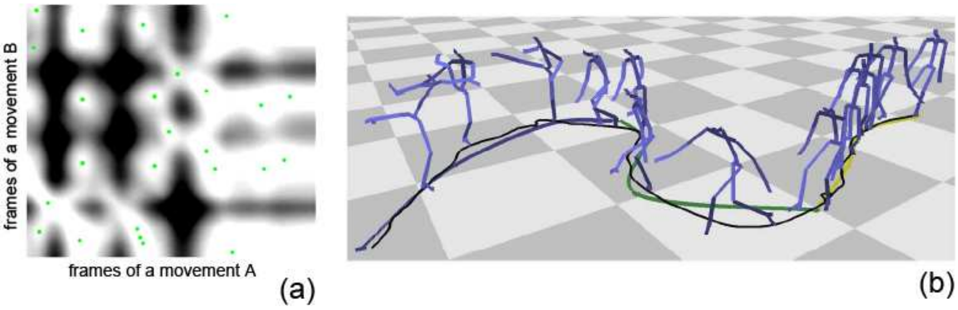


FIG. 2.19 – *Motion Graph* : (a) un exemple de fonction d'erreur pour deux mouvements *A* et *B*, (b) un exemple de changements d'allure tout en suivant une trajectoire.

Ces noeuds sont calculés à l'aide d'une fonction d'erreur (figure 2.20a) : Le point en *(i, j)* représente l'erreur pour faire une transition entre la *i*-ème frame du mouvement A et la *j*-ème frame du mouvement B. Les valeurs blanches correspondent aux basses erreurs et les noires aux hautes. Enfin, les points verts sont les minimums locaux (ie les noeuds à créer).

Comme on peut le voir sur la figure 2.20b, le résultat est plutôt convaincant. Toutefois, si les mouvements à mélanger sont dynamiquement trop différents ou si la base de données est trop limitée, des transitions invalides peuvent être choisies. Pour pallier à ces mauvais choix, Ashraf et al dans [AW01], divisent le squelette humain en deux groupes (haut et bas) et calculent alors deux *Motion graphs*.

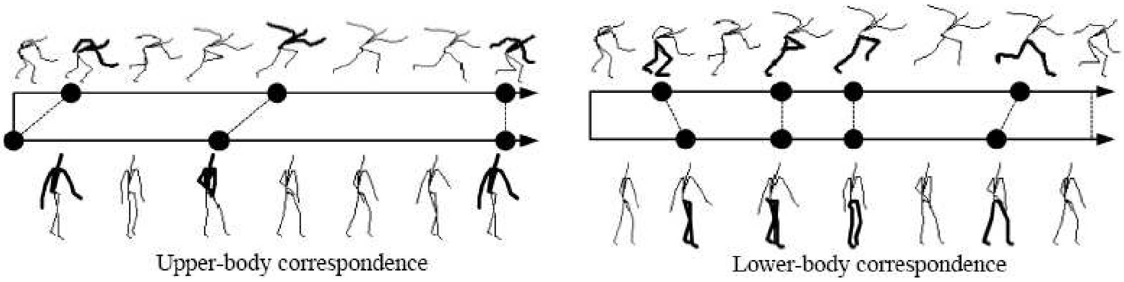


FIG. 2.20 – Séparation du squelette pour calculer les *Motion graphs*.

En dépit de cette amélioration, cette méthode exige souvent l'utilisation de très grandes bases de données afin d'assurer un ensemble minimal de bonnes transitions entre les postures : de telles bases de données ne sont pas appropriées pour des animations interactives. En effet, l'interactivité réclame un grand nombre de configurations posturales. Il est à noter qu'elles ne sont pas non plus appropriées aux animations de foule.

2.3.2 Les quadrupèdes

Même si l'idée du squelette normalisée utilisé dans MKM parait séduisante, il nécessite des modifications afin de pouvoir prendre en compte les différentes morphologies des quadrupèdes. Certaines recherches se sont orientées vers une représentation de ces animaux.

Malgré la grande diversité des espèces dans le monde animal, Revéret et al. dans [RFDC05], ont montré qu'en faisant varier trois paramètres intuitifs (figure 2.21a), on pouvait obtenir un squelette déformable adapté à tous les quadrupèdes : la création d'un squelette générique devient alors très simple. Chaque morphologie peut alors être représentée par la variation de ces paramètres par rapport au squelette de base (figure 2.21b). Toutefois l'utilisation de ce squelette dans MKM^{quad} n'est pas pertinente : l'affectation de mouvements et l'adaptation interactive au sol réclameraient, pour la majorité des articulations, des calculs de cinématique inverse plutôt coûteux. Or, toute la force de MKM repose sur le fait qu'il fonctionne en temps-réel.

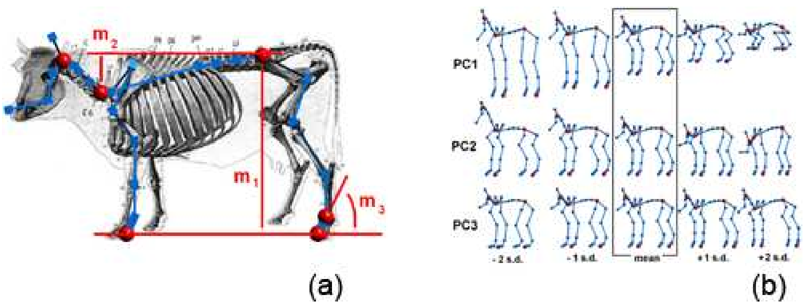


FIG. 2.21 – *Morphable model of quadrupeds skeletons* : (a) les trois paramètres contrôlant le modèle (longueur des jambes, courbure de la colonne vertébrale, mode de locomotion : plantigrade...), (b) variations du modèle en fonction des paramètres.

Cette remarque peut être imputée à la majorité des représentations d'animaux : on décide donc de reprendre le squelette du bipède de MKM et de l'adapter à nos besoins.

Chapitre 3

D'un moteur pour bipèdes à celui d'un moteur pour quadrupèdes

3.1 Le squelette générique de quadrupèdes

Dans MKM, la mise en place d'un squelette générique permet d'adapter un mouvement à n'importe quelle morphologie de bipède et permet de simplifier de nombreux calculs : pour cela, le corps humain est divisé en chaînes cinématiques qui partent de la racine jusqu'aux extrémités.

3.1.1 Un peu d'anatomie...

Pour reproduire un mouvement quadrupède de manière réaliste, il est essentiel de déterminer quelles articulations on doit ajouter à la définition du squelette de bipède. Autrement dit, quelles sont ses articulations caractéristiques. Celles ci sont étroitement liées au mode de locomotion de l'animal.

Les articulations caractéristiques

Dans le cas d'un quadrupède, Szunyoghy et Fehér dans [SF96], distinguent trois types de locomotion :

- plantigrade (figure 3.1a) : se dit d'un animal qui marche sur la plante (métacarpes pour les pattes avant et métatarses pour les pattes arrières) et les doigts (phalange) - (ours).
- digitigrade (figure 3.1b) : se dit d'un animal qui marche sur les doigts - (chien).
- onguligrade (figure 3.1c) : se dit d'un animal qui marche sur le bout des doigts - (cheval).

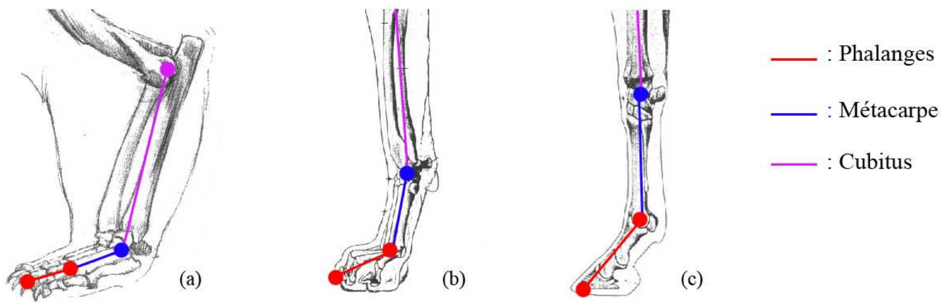


FIG. 3.1 – Anatomie de : (a) une patte d'ours (plantigrade) - (b) une patte de chien (digitigrade) - (c) une patte de cheval (onguligrade).

Les parties des pattes en contact avec le sol sont donc fortement dépendantes du mode de déplacement de l'animal : d'où la nécessité de prendre absolument en compte les métacarpes/métatarses et les phalanges afin de pouvoir représenter fidèlement ces trois *patterns*.

Par la suite, nous retiendrons donc les articulations du squelette représenté par la figure 3.2. Il est à noter que les articulations intermédiaires de la clavicule, entre le buste et les épaules, n'ont aucun degré de liberté, quelque soit le mouvement ou l'animal. Par mesure de simplification, nous avons donc décidé de rattacher les épaules directement au buste.

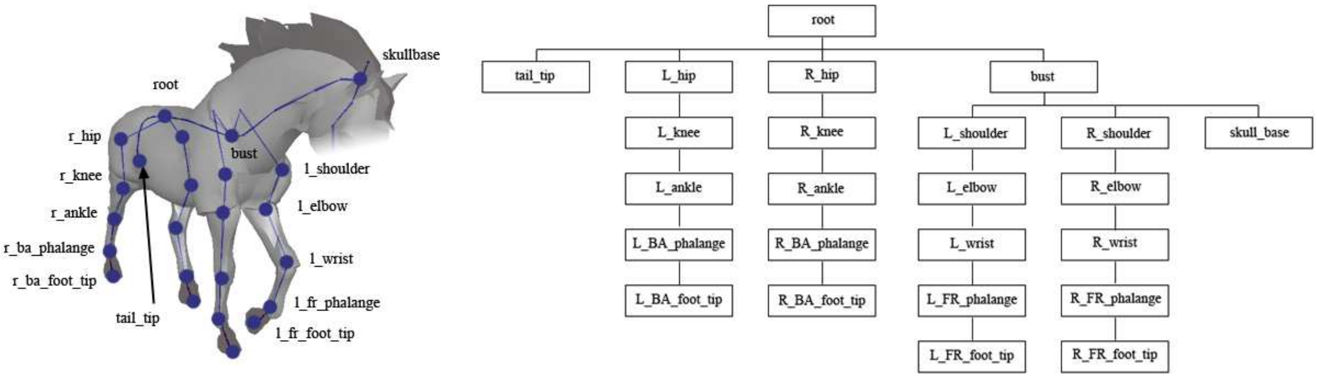


FIG. 3.2 – Représentation des articulations obligatoires pour un quadrupède.

Maintenant que nous avons isolé les articulations essentielles, il est nécessaire d'en étudier les particularités afin d'en extraire une représentation normalisée.

Les caractéristiques de ces articulations

Cette étude s'est faite par le biais de planches anatomiques [SF96], de photos [Muy57] et d'animations de mouvements de différents quadrupèdes : elle n'est donc pas exhaustive. De plus, il existe beaucoup moins d'informations et de connaissances pour les animaux que pour les humains. Enfin, la grande diversité des espèces existantes implique qu'il existe évidemment des cas particuliers ne vérifiant pas certaines règles.

Il faut donc choisir un juste compromis entre réalisme et simplification des calculs. Nous parlons de *réalisme*, si le mouvement est validé par un échantillon de personnes ayant des notions d'animation et/ou d'anatomie. Notre but n'est donc pas de refléter à tout prix la réalité, mais de proposer des solutions cohérentes.

Chez un quadrupède, il est nécessaire de comprendre le fonctionnement des articulations constituant les pattes, pour pouvoir représenter ses mouvements de locomotion. D'après des calculs effectués sur les animations présentes au laboratoire, ainsi que certaines planches anatomiques, on s'apercoit que les articulations qui constituent une patte ne sont pas planaires entre elles. De plus, même si elles n'ont qu'un degré de liberté chacune, elles ne se déplacent pas sur des plans parrallèles.

La représentation existante du squelette générique ne s'avère donc pas adaptée.

3.1.2 Les limites des représentations existantes

Rappelons que pour les bipèdes les membres sont représentes par deux articulations, la hanche et la cheville pour les jambes, l'épaule et le poignet pour les bras. Ces derniers sont ensuite reliés à la colonne par des segments normalisés. Le genou et le coude sont alors calculés de manière simple à l'aide de sphères.

Comme nous l'avons démontré précédemment, ces informations sont loin d'être suffisantes pour capturer les caractéristiques d'un mouvement quadrupède : il manque plusieurs articulations pour les pattes.

Pour remédier à cela, une première solution consisterait à relier par des liens normalisés les articulations manquantes. Mais, en plus de provoquer des calculs supplémentaires pour prendre chaque articulation en compte, on se heurterait à un manque de réalisme pour l'adaptation des pattes au sol. En effet, ces nouvelles articulations ne s'adapteraient pas à un sol non plat, on aurait ainsi une impression de rigidité sur la fin des pattes.

On pourrait alors choisir une articulation de fin différente pour représenter les membres. Cela impliquerait alors le calcul de plus d'une articulation intermédiaire, sans quoi l'on perdrait des articulations et donc, des caractéristiques du mouvement très importantes pour le réalisme de l'animation. Or, ces calculs ne pourraient plus être aussi simple que pour les bipèdes. Il faut alors éviter d'avoir recours à des calculs de cinématique inverse plutôt coûteux. Ceci est possible en utilisant une représentation multi-résolution, en supplément des sphères et des plans.

Nous proposons donc de reprendre la démarche utilisée pour le squelette adimensionnel de MKM. Tout en essayant de couvrir l'ensemble des espèces quadrupèdes, ce modèle permet de respecter la contrainte d'animer plusieurs animaux en temps-réel.

3.1.3 La solution choisie

Afin d'éviter tout calcul de cinématique inverse avec inversion de matrice jacobienne, il est nécessaire que les pattes soient représentées par un seul segment de longueur variable et que les articulations intermédiaires soient rapidement calculables de manière analytique.

La solution choisie est donc de garder la division en chaînes cinématiques : on représente ainsi la colonne vertébrale et la queue, par une même spline passant par les points de contrôle placés à la base de la tête, le cou, le pelvis et l'extrémité de la queue. Pour chaque patte, on utilise un segment normalisé de longueur variable reliant 2 articulations : l'épaule, l'extrémité de la patte pour l'avant et la hanche, l'extrémité de la patte pour l'arrière (figure 3.3b). On attache également à chaque patte, un système multi-résolution qui est détaillé un peu plus bas. Enfin, les liens entre les pattes et la colonne vertébrale sont représentés par des segments normalisés de tailles fixes.

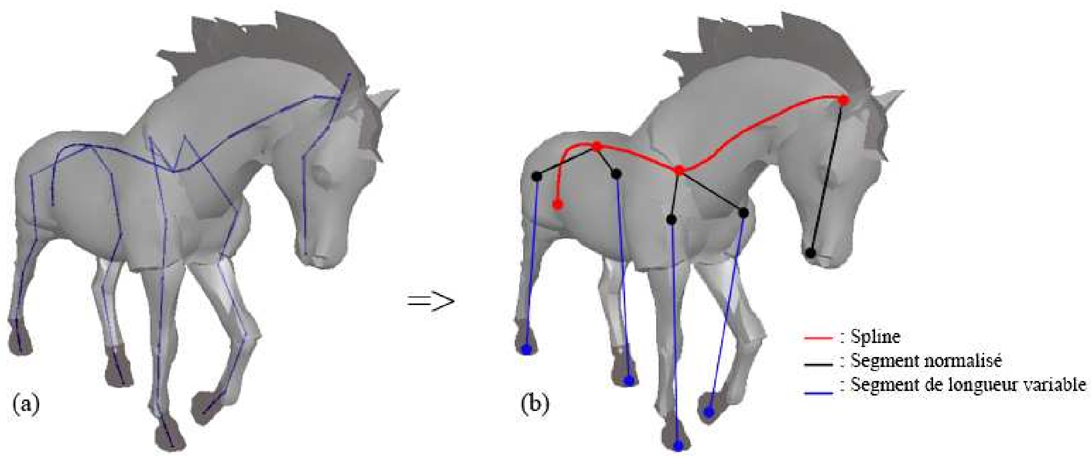


FIG. 3.3 – (a) Squelette d'animation d'un cheval - (b) Squelette générique pour un quadrupède dans MKM^{quad}

La représentation des pattes

Grâce à un système multi-résolution, il est possible de conserver la représentation des pattes par un segment normalisé de taille variable et de retrouver de manière analytique et rapide, la position de plusieurs articulations intermédiaires.

Comme nous l'avons vu précédemment, les articulations intermédiaires qui composent la patte d'un quadrupède n'ont qu'un degré de liberté mais ne sont pas planaires entre elles. Il est donc nécessaire de définir autant de plans que d'articulations à calculer (figure 3.4a).

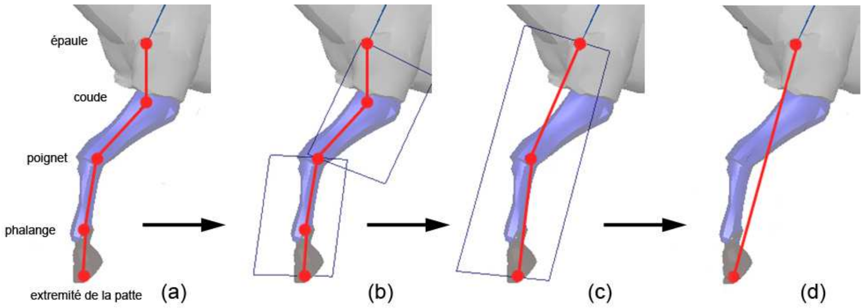


FIG. 3.4 – Encodage des articulations intermédiaires pour les pattes d'un quadrupède.

Dans un premier temps, on considère la patte d'un animal comme la concaténation de deux segments de taille variable (figure 3.4b) : l'un qui part de l'épaule au poignet et l'autre du poignet à l'extrémité de la patte (pour une patte avant). Comme pour les segments normalisés de taille variable du squelette bipède, on accroche à sa racine un plan de référence et on lui attribue un scalaire qui correspond au pourcentage de la longueur maximum de celui-ci.

On se retrouve alors dans la configuration d'un membre d'humain avec une seule articulation intermédiaire (figure 3.4c), à la différence près que les segments de corps qui constituent la patte sont de longueurs variables. On réitère donc l'opération de normalisation grâce à un nouveau segment de taille variable qui part de l'épaule à l'extrémité de la patte (figure 3.4d) .

Le fait d'utiliser un plan pour chaque articulation intermédiaire ainsi qu'un raffinement multi-résolution, permet alors de calculer, sans cinématique inverse coûteuse et sans erreurs, toutes les configurations de la patte d'un quadrupède.

Le calcul des articulations intermédiaires

Le dernier module du moteur d'animation, permet adapter le squelette générique morphé à l'environnement dynamique. Une fois adapté, il calcule les articulations intermédiaires (genoux, chevilles...) afin de pouvoir visualiser correctement le mouvement. Grâce à la représentation générique que nous avons choisit pour nos animaux quadrupèdes, retrouver ces articulations se fait de manière rapide et simple et n'altère pas l'interaction de la plate-forme.

Le système multi-résolution utilisé pour représenter les pattes, ainsi que les points communs que nous avons observé sur l'anatomie des bipèdes, nous permet de raffiner le niveau de détail d'une patte en calculant simplement des intersections plan-sphères.

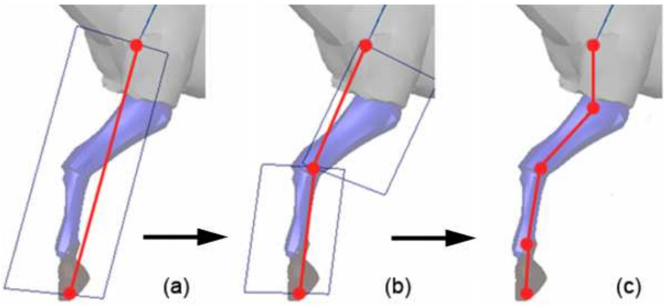


FIG. 3.5 – Décodage des articulations intermédiaires pour les pattes d'un quadrupède.

Soit $s1$, $s2$ et $s3$, les scalaires stockés dans le fichier de mouvement pour une patte avant correspondant respectivement aux segments de taille variable épaule-poignet, poignet-extremité de la patte et épaule extremité de la patte.

On calcule tout d'abord l'articulation du poignet. Celle-ci est placée à l'intersection de deux sphères. La première est centrée sur l'épaule et son rayon est égal à la taille de l'humérus et du cubitus multipliée par le scalaire $s1$. La seconde est centrée sur l'extremité de la patte et son rayon est égal à la taille du métacarpe et des phalanges multipliée par le scalaire s2. Le poignet est alors placé à l'intersection du cercle et du plan de référence, dont l'orientation est stockée dans le fichier S4D.

De plus, comme le plan est définit dans le repère de l'épaule, la position relative du poignet est facilement calculable :

$$limb = (humerus + cubitus + metacarpe + phalange) \times s3$$

$$z = \frac{((cubitus + humerus) \times s1)^2 - ((metacarpe + phalange) \times s2)^2 + limb^2}{2 \times limb}$$

$$y = \sqrt{arm^2 - z^2}$$

où z et y sont les coordonnées locales du poignet dans le plan de l'épaule (figure 2.6), $humerus$, $cubitus$, $metacarpe$ et $phalange$ respectivement la longueur de l'humérus, du cubitus, du métacarpe et de la phalange et $limb$ la longueur courante entre l'épaule et l'extremité de la patte.

Maintenant que l'on a l'articulation du poignet (figure 3.5b), on peut aisément retrouver les articulations du coude et des phalanges par la même technique que pour les bipèdes. Le calcul des articulations des pattes arrières se fait de la même manière aussi, étant donné que leur stockage est identique à celui des pattes avant.

Par ce système multi-résolution, nous avons donc définit un nouveau modèle normalisé de quadrupèdes pour encoder un mouvement de manière adimensionnelle. Maintenant il est nécessaire d'associer à ce mouvement un ensemble de contraintes.

3.2 La définition de contraintes

De la même manière que pour un bipède, il est nécessaire d'associer des contraintes aux mouvements quadrupèdes. Ces contraintes sont essentiellement de type "phases de support" afin de permettre la synchronisation de mouvements.

Comme nous l'avons montré précédemment, ce support peut être de plusieurs types pour un animal et influe beaucoup sur son mode de déplacement. Par exemple, un ours (plantigrade), dans un mouvement de locomotion, doit avoir les phalanges et les métacarpes en contact avec le sol lors de la phase de support d'une de ses pattes. Pour éviter tout glissement sur le sol d'un os qui devrait être en contact avec le sol, il est donc nécessaire d'ajouter au mouvement une information sur le *pattern* de locomotion.

Nous décidons donc de garder le principe de calcul semi-automatique déjà existant. Pour rappel, ce calcul se fait grâce au réglage à deux paramètres : la hauteur maximale h autorisée pour l'extrémité la plus basse du pied et la vitesse maximale autorisée quand le pied est en dessous du seuil précédent.

On peut alors, à partir de ces deux paramètres, détecter automatiquement le type de locomotion du modèle traîté. Pour cela, il est nécessaire d'étudier la position des articulations des phalanges et des poignets. Si les deux se trouvent sous le seuil h lors de la détection d'une phase de support, on en conclue alors que le modèle est plantigrade. Si seule l'articulation des phalanges se trouve sous le seuil, on peut en conclure que le modèle est digitigrade. Dans les autres cas, le modèle est onguligrade.

Cette donnée est ajoutée au fichier S4D et est invariante pour l'ensemble du mouvement. Toutefois, malgré la detection automatique, l'utilisateur peut spécifier lui-même le type de locomotion de l'animal, afin de rendre la contrainte de support moins restrictive par exemple.

3.3 Une nouvelle approche de la synchronisation

Comme nous l'avons vu, la relation algébrique utilisée dans MKM n'est pas extensible aux quadrupèdes car le théorème associé ne garantit pas la présence d'au moins une solution en cas d'erreur. De plus, les autres recherches menées dans le domaine de la synchronisation ne sont pas adaptées à notre cas ou ne sont pas optimales. Il a donc été nécessaire de trouver une nouvelle approche.

3.3.1 Nouvelle approche

Pour cela, nous avons décidé d'étudier au niveau anatomique, la manière dont un quadrupède effectue des transitions. Dans [How44], Brazier Howell montre qu'il n'existe que neuf allures différentes chez tous les quadrupèdes : celles-ci sont ensuite modulées par l'espèce de l'animal et son style, mais la séquence des pieds d'appui est toujours la même. Grâce à ces allures, l'animal peut changer sa vitesse de déplacement.

A partir de ces informations et des contraintes associées à chaque mouvement, il est donc possible de facilement détecter le type de l'allure et la vitesse de l'animal. Nous proposons ensuite de représenter cette allure sous la forme d'un automate cyclique déterministe orienté (figure 3.6a et b).

3.3.2 Cas particulier

Dans un premier temps, nous décidons d'étudier un cas particulier d'un changement d'allure pour en extraire les informations caractéristiques. Au cours de sa thèse [Fav06] a developpé une application qui permet d'extraire automatiquement les phases de support depuis une vidéo. Ainsi, en plus de l'observation vidéo, nous pouvons analyser l'enchainement des phases de support (figure 3.7).

Visuellement, on peut alors repérer les motifs des deux allures : la marche se distingue par un motif en forme d'escalier, alors que le galop a plutôt un motif en forme de *dents de scie*. . Au vue de la durée des phases de support de plus en plus courte dans le motif de marche, on se rend compte que celle-ci s'accèlère progressivement : comme nous l'avons dit plus haut, chaque allure permet à un animal de se déplacer à une vitesse différente. Pour passer d'une marche à un galop, l'animal va donc accélerer la vitesse de son

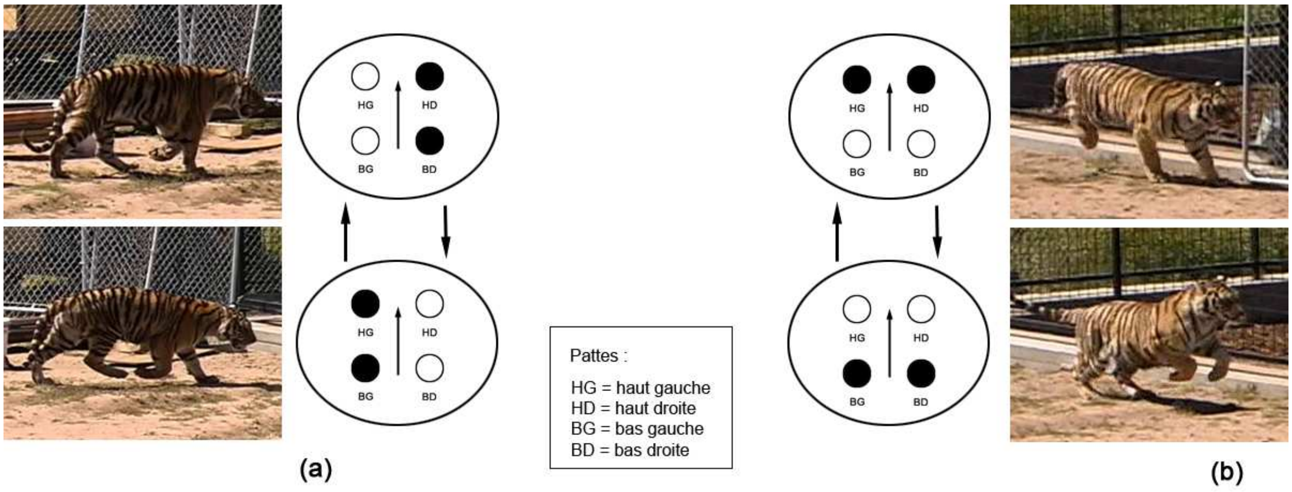


FIG. 3.6 – Représentation d'allure sous forme d'automate déterministe orienté : (a) pour une marche, (b) pour un galop. Les ronds pleins représente une patte au sol et les ronds vides représentent une patte levée.

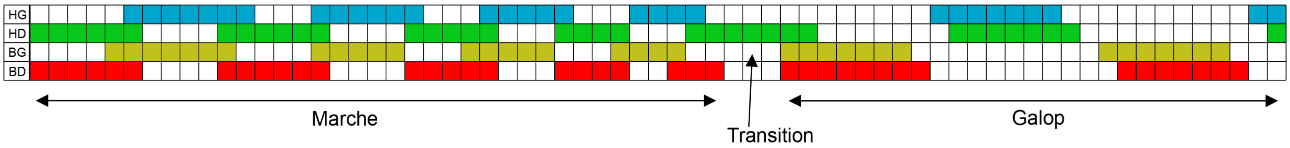


FIG. 3.7 – Exemple de phases de support lors d'un changement d'allure marche-galop.

allure initiale jusqu'à atteindre la vitesse de son allure finale. Comme il ne peut pas garder cette vitesse avec son allure initiale, il créé un point de cassure dans son rythme pour changer d'allure.

Dans le cas d'un passage d'une allure plus lente à une allure plus rapide, le point de cassure se fait par un appui sur la patte avant de l'animal. Ce point de cassure est bien visible sur la figure 3.7 : on voit que l'animal se retrouve avec une phase d'appui unipodal sur la patte avant droite. Ce point de cassure lui permet de s'affranchir du motif de son allure précédente et d'en enchaîner alors un nouveau (figure 3.8).

Pour atteindre la configuration de cassure (sur une patte), l'animal va donc changer sortir de l'automate de son motif initial en levant les pattes inutiles (la patte arrière-droite dans la figure 3.8). Ensuite, pour rejoindre la configuration de la nouvelle allure, il va poser, en plus de la patte avant, les pattes présentes dans l'état d'entrée de l'automate du motif final (les pattes arrière dans la figure 3.8).

Dans la figure 3.8, seul un état de l'automate de marche permet de rejoindre l'automate de galop. Il est à noter que les deux états le peuvent en suivant le même schéma que presenté dans le paragraphe précédent. La patte d'appui de cassure sera par contre différente. Il est aussi important de noter, que pour le passage de la marche au galop, il n'y a qu'un état d'entrée dans l'automate de galop : cet état correspond à la phase d'impulsion (pattes arrières au sol sur la figure 3.6).

Respectivement, le passage du galop à la marche implique qu'il n'y a qu'un état de sortie dans l'automate de l'allure plus rapide : cet état correspond à la phase de réception (pattes arrières au sol sur la figure 3.6). L'automate de marche aura lui, par contre deux états d'entrée.

A partir de ces remarques et de plusieurs observations similaires sur d'autres vidéos, nous pouvons alors

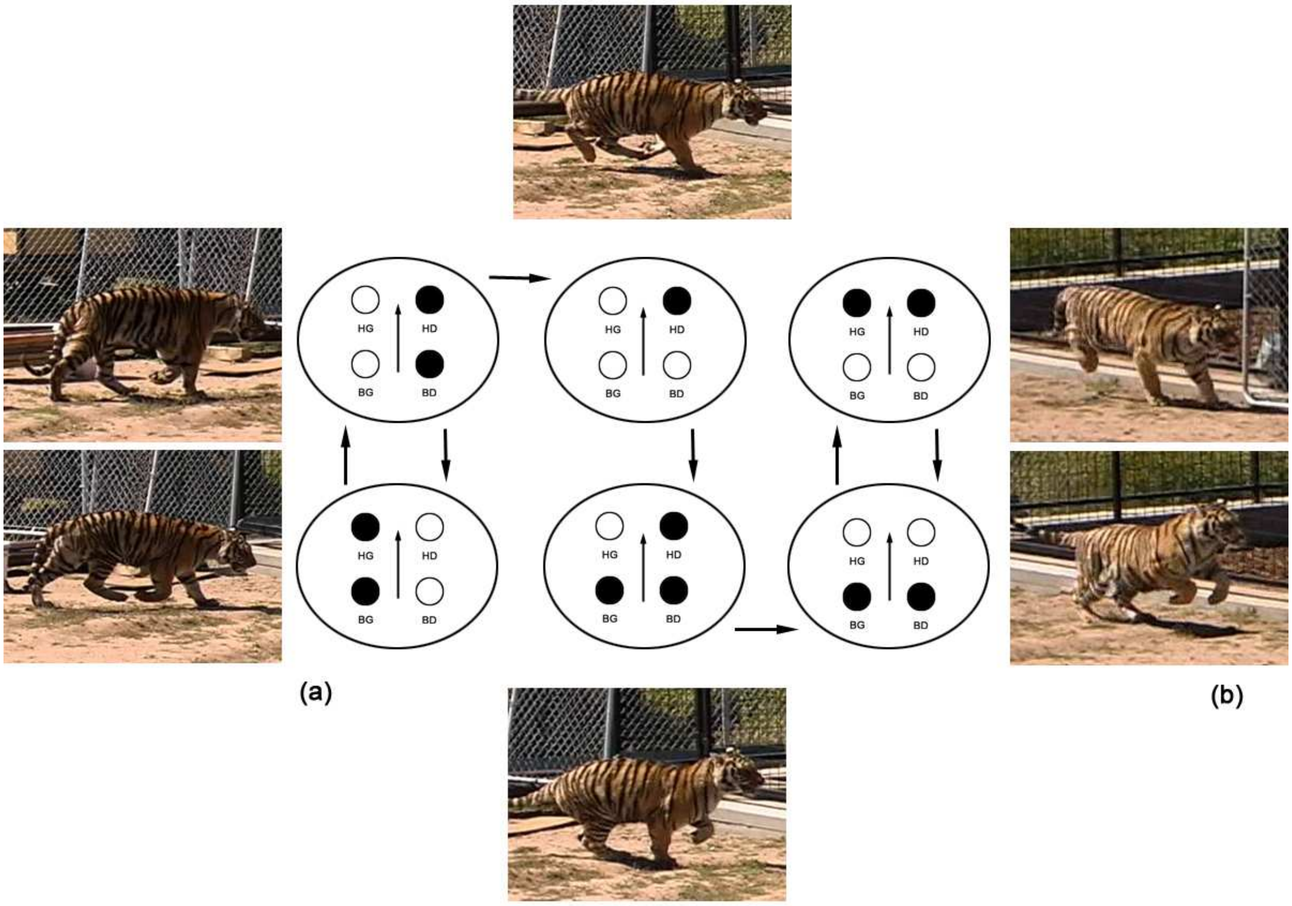


FIG. 3.8 – Représentation d'un changement d'allure marche-galop sous forme d'automate déterministe orienté.

généraliser une méthode de génération de transitions d'une allure moins rapide à une allure plus rapide, puis d'une allure plus rapide à une allure moins rapide.

3.3.3 Généralisation

L'ensemble de ces observations permet alors de décrire une algèbre simple de génération d'états de transitions entre deux allures : dans un premier temps, elle vérifie si on peut générer une transition entre deux états $E1$ et $E2$ appartenant respectivement à deux automates d'allures $A1$ et $A2$. Si cette transition est possible, deux nouveaux états transitifs $T1$ et $T2$ sont créés.

Pour ces deux étapes, on introduit deux nouvelles relations algébriques (figure 3.9a et b) :

- une transition entre $E1$ et $E2$ est possible si l'allure représentée par $A1$ est plus lente que celle représentée par $A2$ et que $E1 - E2$ (relation de la figure 3.9) contient une patte antérieure posée.
- une transition entre $E1$ et $E2$ est possible si l'allure représentée par $A1$ est plus rapide que celle représentée par $A2$ et que $E1 - E2$ (relation de la figure 3.9) contient une patte postérieure posée.
- si une transition est possible entre $E1$ et $E2$, $T1 = E1 - E2$ et $T2 = T1 + E2$.

Cette généralisation nous permet alors, à partir de deux allures et de leurs vitesses, de générer automatiquement une séquence de transitions (figure 3.10) : on économise ainsi le passage dans le module de mélange de mouvements.

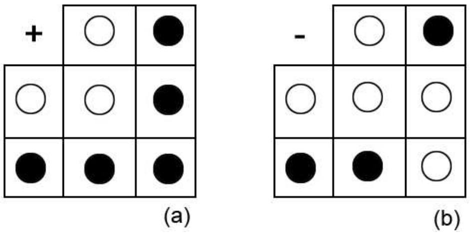


FIG. 3.9 – Allure des relations algébriques + et - qui permettent de déterminer si une transition est possible entre deux états appartenant à deux automates d'allure. Si c'est possible, elles permettent de calculer les deux états transitifs

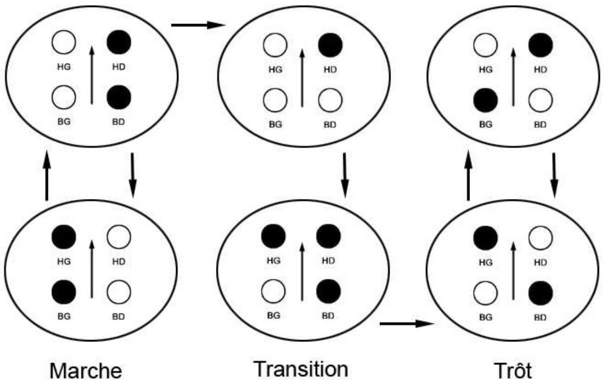


FIG. 3.10 – Exemple de calcul d'une transition entre une allure de marche et une allure de trôt à partir de notre algèbre.

Il est à noter que nous ne disposons pas encore au laboratoire de plusieurs allures pour un même animal et, de ce fait, la généralisation n'a pas pu être visuellement vérifiée. Toutefois, elle a été soumis à des experts en anatomie animale du muséum d'Histoire Naturelle de Paris qui ont approuvé la plausibilité de la méthode.

La séquence de transition générée permet de passer d'un motif à un autre de manière plausible. Pour interpoler la vitesse, nous retiendrons la solution implémentée dans le moteur MKM et utilisée lors de l'allongement d'une phase de support pour un bipède. Les distances et les durée de chaque état de transition est aussi interpolé entre l'automate d'allure original et l'automate d'allure terminal.

Enfin, on garde le même principe pour le module d'adaptation du squelette au sol : on considère alors l'animal comme l'union de deux bipèdes à adapter. Soit $h1$ et $h2$ la hauteur de chaque patte avant, $h3$ et $h4$ la hauteur de chaque patte arrière, et $h5$ et $h5$ respectivement la hauteur du buste et de la racine : on aura alors $hBust = min(h1, h2, h5)$ et $hRoot = min(h3, h4, h6)$.

Chapitre 4

Résultats et tests

Toutes les versions originales de squelette des exemples suivants ont été réalisés par Christine Depraz sous Maya 7.0.

4.1 Calcul du squelette générique

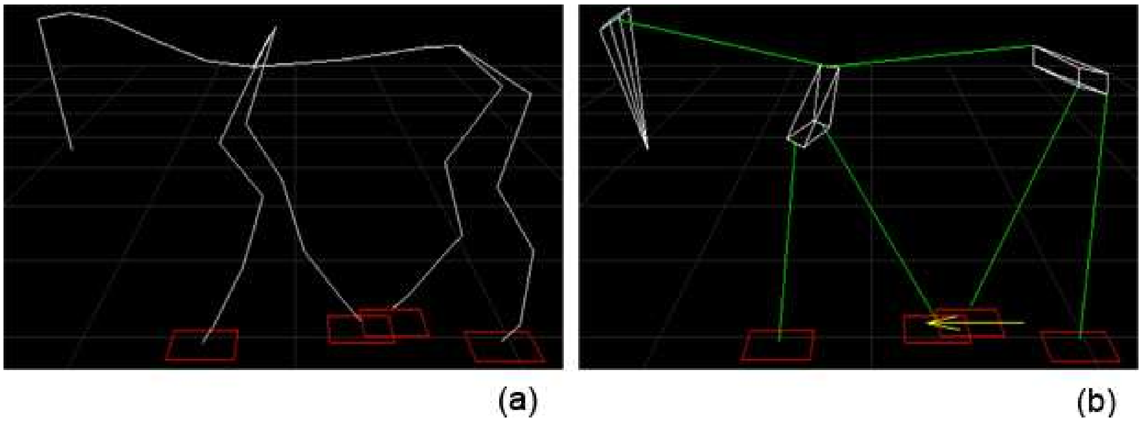


FIG. 4.1 – Calcul du squelette générique : (a) Squelette de cheval original, (b) Squelette normalisé.

A partir d'un squelette d'animation animée (figure 4.1a), nous pouvons calculer un squelette générique (figure 4.1b) avec lequel on encode le mouvement quadrupède. Les polygones blancs correspondent aux segments normalisés de taille fixe tandis que les traits verts correspondent à la spline de la colonne vertébrale et aux segments de longueur variable des pattes.

4.2 Reconstruction à partir d'un squelette générique

Ce squelette générique (figure 4.2a) nous permet alors de reconstruire un squelette à partir de données morphologiques (figure 4.1b). Comme l'illustre la figure 4.3, cette reconstruction se fait sans perte d'informations. Les petites différences au niveau du cou sont seulement dûs au fait que la spline n'est pas suffisament raffinée.

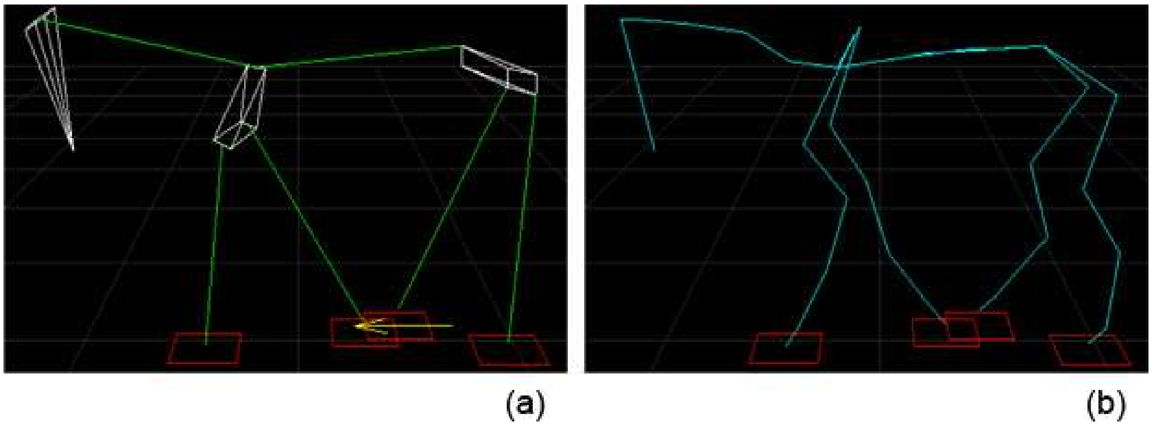


FIG. 4.2 – Reconstruction à partir d'un squelette générique : (a) Squelette de cheval original, (b) Calcul du squelette générique, (c) Reconstruction à partir du squelette générique.

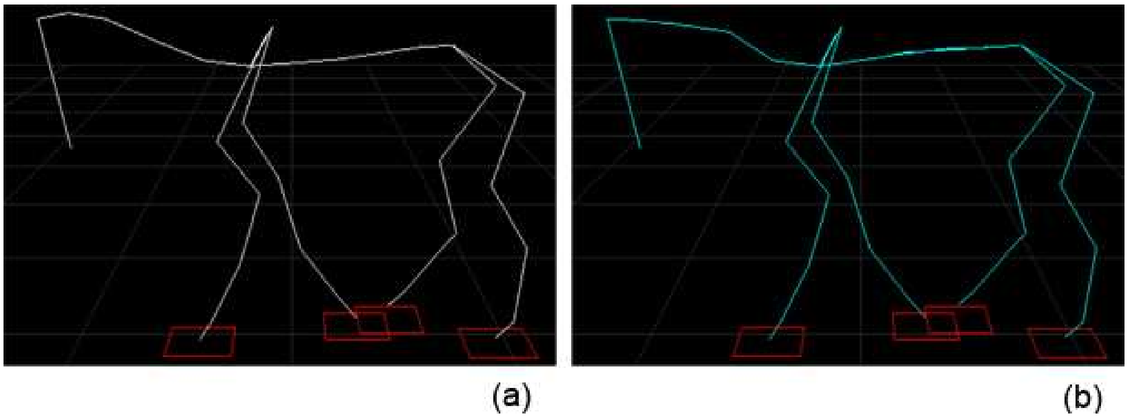


FIG. 4.3 – Comparaison entre le squelette original (a) et le squelette reconstruit (b) à partir du squelette normalisé.

4.3 Ajout de contraintes de footsteps

On peut ajouter au mouvement, un ensemble de contraintes de phases de support qui sont détectées de manière semi-automatique. L'utilisateur renseigne une hauteur maximale et une vitesse (figure 4.4).

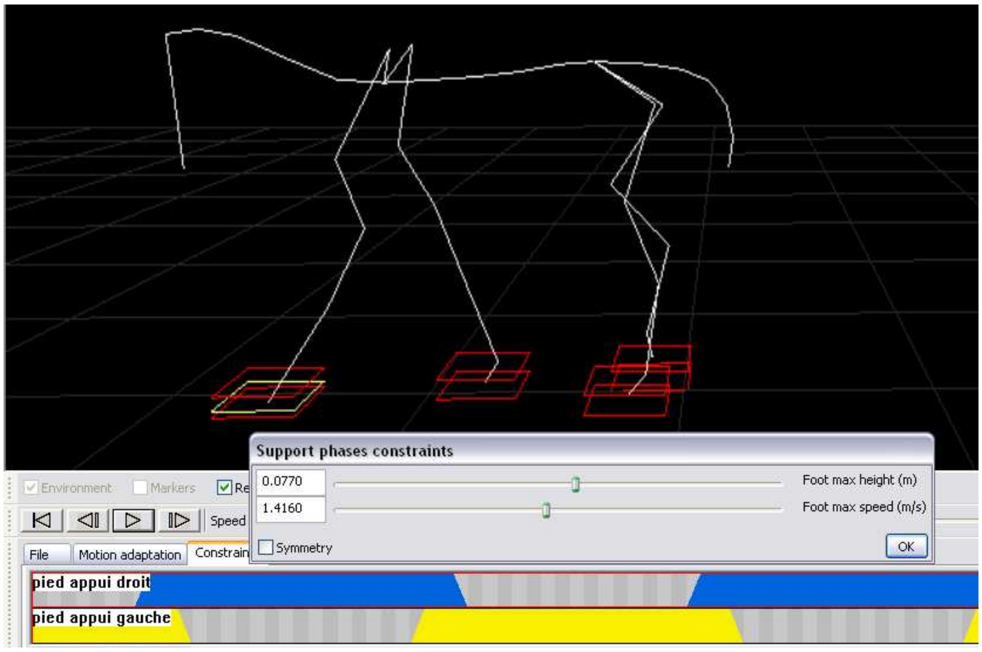


FIG. 4.4 – Comparaison entre le squelette original (a) et le squelette reconstruit (b) à partir du squelette normalisé.

Chapitre 5

Conclusion

Dans ce rapport, nous avons proposé une adaptation aux quadrupèdes, d'un moteur d'animation de bipèdes. Cette adaptation respecte l'ensemble des contraintes du moteur original : elle permet, en temps-réel, de mélanger différents mouvements et de les affecter simultanément à plusieurs quadrupèdes de morphologies différentes, évoluant dans un environnement interactif.

Pour cela, les mouvements sont encodés à l'aide de notre nouvelle représentation normalisée du squelette d'animation de quadrupède. Grâce à cette représentation, les données ne dépendent plus de la morphologie de l'animal de base et peuvent donc être affectées à d'autres animaux de manière rapide. Les positions des articulations intermédiaires des pattes (coude, poignet, phalange) étant fortement dépendantes de la morphologie de l'animal, ces dernières ne sont pas encodées.
Nous proposons donc de les représenter à l'aide d'un système multi-résolution : de cette manière, elles peuvent ensuite être calculées de manière analytique. On évite alors de passer par de lourds calculs de cinématique inverse, pourtant souvent présents dans les moteurs d'animation classiques.

De plus, nous proposons d'associer aux mouvements un ensemble de contraintes de phases de support et une donnée sur le *pattern* de locomotion de l'animal (plantigrade, digitigrade, onguligrade). Grâce aux phases de support, nous évitons ainsi de mélanger deux postures incompatibles. Grâce au *pattern*, nous évitons également les artefacts classiques d'un moteur d'animation comme le glissement des pattes sur le sol.

Enfin, grâce à une observation rigoureuse des transitions d'allure chez les quadrupèdes, nous avons proposé une nouvelle méthode pour synchroniser plusieurs mouvements de quadrupèdes en temps-réel. En exprimant les mouvements sous la forme d'automates déterministes finis, notre méthode permet de calculer les phases de support transitoires entre deux états appartenant à des automates (et donc des mouvements) différents. Cette méthode se base sur des paramètres simples, comme la vitesse des mouvements à mélanger, pour calculer la transition.

Étant donné la grande diversité des animaux et le peu d'informations que nous avons, nous admettons que notre module de transition ne génère pas, pour toutes les espèces, des transitions collant parfaitement à la réalité. Toutefois, il est à noter que ces dernières sont dans tous les cas plausibles.

Chapitre 6

Perspectives

Dans de récents travaux en anatomie, Anick Abourachid dans [Abo03], a proposé une approche théorique complète qui permet de quantifier toutes les coordinations locomotrices, en allures constantes comme dans les transitions : les mouvements locomoteurs s'inscrivent alors dans une séquence qui débute par les posés des membres antérieurs suivis de ceux des membres postérieurs. La locomotion est alors considérée comme une succession d'actions définies dans deux dimensions : le temps et l'espace.

Elle propose donc une table théorique qui identifie toutes les allures possibles en tenant compte de trois paramètres : le décalage temporel entre les posés des deux pattes antérieures, des deux pattes postérieures et entre ces deux paires de pattes. Ces paramètres sont représentés par des intervalles afin de représenter tous les quadrupèdes.

Ludovic Maes, dans [Mae06], apporte des données précises et la vérification de la table théorique des allures pour l'exemple des chiens et propose, à son tour, une table théorique qui identifie les allures possible en tenant compte de trois paramètres : le décalage spatial entre les posés des deux pattes antérieures, des deux pattes postérieures et entre ces deux paires de pattes. Ici aussi, ces paramètres sont représentés par des intervalles afin de représenter tous les quadrupèdes. Les recherches s'orientent maintenant à trouver les valeurs exactes de ces données selon l'animal.

Lionel Revéret et al. dans [RFDC05], ont montré qu'en faisant varier trois paramètres intuitifs (figure 2.21a), on pouvait obtenir un squelette déformable adapté à tous les quadrupèdes.

Le but serait donc de coupler ces trois découvertes afin de proposer un modèle haut-niveau d'animation de quadrupèdes. L'idée est de trouver une relation en les données des tableaux théoriques spatiaux et temporels et les trois paramètres du squelette déformable de quadrupèdes. Ces paramètres sont ensuite facilement récupérables sur un squelette de MKM^{quad} et permettraient ainsi de calculer une table avec les données spatiales et temporelles.

A partir de ces tableaux, et de données précises, on peux alors pour n'importe quel quadrupède, générer automatiquement une allure et calculer des transitions réalistes entre ces allures.

On pourrait alors utiliser des coefficients de squelette pour générer des squelettes de dinosaures, par exemple, et simuler leurs locomotions en temps-réel et dans un environnement interactif.

Bibliographie

[Abo03] Anick Abourachid. A new way of analysing symmetrical and asymmetrical gaits in quadrupeds. In *Comptes rendus de Biologie*, Mars 2003.

[AW01] Golam Ashraf and Kok Cheong Wong. Constrained framespace interpolation. In *Computer Animation 2001*, pages 61–72, November 2001.

[BW95] Armin Bruderlin and Lance Williams. Motion signal processing. In *Proceedings of SIGGRAPH 1995*, pages 97–104, August 1995.

[CLS02] Matt Cartmill, Pierre Lemelin, and Dan Schmitt. Support polygons and symmetrical gaits in mammals. In *Zoological Journal of the Linnean Society*, volume 136, pages 401–420, 2002.

[Fav06] Laurent Favreau. *Capture, modélisation et animation du mouvement animal à partir de la vidéo*. PhD thesis, Institut National Polytechnique de Grenoble, 2006. En cours de rédaction.

[GRG94] Shang Guo, James Robergé, and Thom Grace. Controlling movement using parametric frame space interpolation. In *Models and Techniques in Computer Animation*, pages 216–227, 1994.

[How44] Brazier Howell. *Speed in animals*. University of Chicago, 1944.

[IC87] Paul Isaacs and Michael Cohen. Controlling dynamic simulation with kinematic constraints. In *Proceedings of SIGGRAPH 1987*, pages 215–224, July 1987.

[KGP02] Lucas Kovar, Michael Gleicher, and Frederic Pighin. Motions graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques 2002*, volume 1, pages 473–482, 2002.

[KM05] Richard Kulpa and Franck Multon. Fast inverse kinematics and kinetics solver for human-like. 2005.

[KMA05] Richard Kulpa, Franck Multon, and Bruno Arnaldi. Morphology-independent representation of motions for interactive human-like animation. In *Computer Graphics Forum 2005*, volume 3, 2005.

[LP02] Karen Liu and Zoran Popovic. Synthesis of complex dynamic character motion from simple animations. In *Proceedings of SIGGRAPH 2002*, pages 408–416, July 2002.

[LvdPF00] Joe Laszlo, Michiel van de Panne, and Eugene Fiume. Interactive control for physically-based animation. In *Proceedings of SIGGRAPH 2000*, pages 201–209, July 2000.

[Mae06] Ludovic Maes. Blabla. Master's thesis, Université de Biologie, Paris, 2006.

[MKM] Manageable kinematic motions.

[MKMA04] Stéphane Ménardais, Richard Kulpa, Franck Multon, and Bruno Arnaldi. Synchronisation for dynamic blending of motions. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2004*, volume 3, pages 325–336, August 2004.

[MMKA04] Stéphane Ménardais, Franck Multon, Richard Kulpa, and Bruno Arnaldi. Motion blending for real-time animation while accounting for the environment. In *Computer Graphics International 2004*, volume 8, June 2004.

[Mén03] Stéphane Ménardais. *Fusion et adaptation temps réel de mouvements acquis pour l'animation d'humanoïdes synthétiques*. PhD thesis, Université Rennes 1, January 2003.

[Muy57] Eadweard Muybridge. *Animals in motions*. Dover Pictorial Archive Series. Dover Publications, Inc, 31 East 2nd Street, Mineola, N.Y. 11501, 1957.

[RFDC05] Lionel Reveret, Laurent Favreau, Christine Depraz, and Marie-Paule Cani. Morphable model of quadrupeds skeletons for animating 3d animals. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005*, July 2005.

[SF96] Andas Szunyoghy and György Fehér. *Grand cours d'anatomie artistique*. Könemann, Bonner Str. 126, D-50968 Köln, 1996.

[TGB00] Deepack Tolani, Ambarish Goswami, and Normal Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. In *Graphical Models*, volume 5, pages 353–388, September 2000.

[Zel82] David Zelter. Motor control techniques for figure animation. In *IEEE Computer Graphics and Applications*, pages 53–59, November 1982.

[ZSC90] Z Zatsiorsky, V Seluyanov, and LG Chugunova. Methods of determining mass-inertial characteristics of human body segments. In *Contemporary problems of Biomechanics*, pages 273–291, 1990.