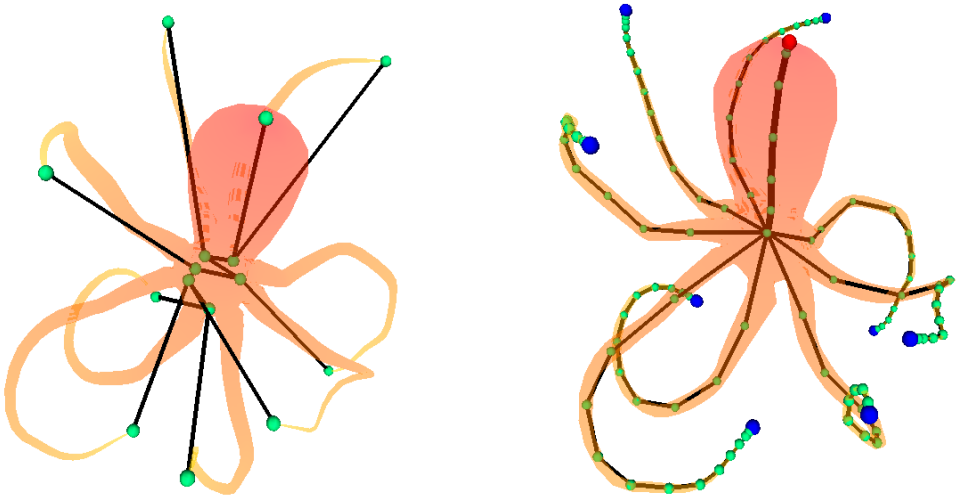
D'un squelette géométrique à un squelette d'animation

Grégoire Aujay

D'un squelette géométrique à un squelette d'animation

Master Recherche Image Vision Robotique
Promotion 2005/2006, Mars-Juin 2006



Grégoire AUJAY
Superviseurs : Franck Hetroy et Francis Lazarus
EVASION / GRAVIR IMAG INRIA
INRIA Rhône-Alpes - ZIRST
655 avenue de l'Europe, 38334 Saint Ismier Cedex

Table des matières

Chapitre 1

Contexte

1.1 Le problème posé

Pour l'animation en particulier et plus généralement pour toute application qui nécessite de caractériser un modèle 3D par sa forme, on cherche une structure hiérarchique qui décrit le mieux possible la morphologie des objets. L'animation requiert un squelette ayant la structure d'un graphe dont les nœuds sont considérés comme des articulations et les arêtes comme des os. Ce squelette doit capturer au mieux la forme de l'objet, pour correspondre à un squelette au sens anatomique. Un tel squelette peut ensuite être utilisé pour animer l'objet mais aussi servir à trouver des propriétés de la forme 3D ou à la comparer à une autre en vue de la ranger dans une base de données par exemple. Si pour certaines applications seule la topologie du squelette compte, en animation une contrainte supplémentaire oblige à plonger le squelette, c'est à dire le positionner correctement par rapport à la forme 3D, tout comme un squelette anatomique est plongé dans un corps.

Construire un squelette d'animation pour un modèle 3D est une tâche longue et fastidieuse où chaque articulation doit être placée manuellement. Une construction automatique ou guidée permettrait un gain de temps considérable, surtout pour un projet nécessitant la présence de nombreux modèles animés. Nous cherchons donc à automatiser ce processus mais en laissant la possibilité à l'utilisateur, s'il le souhaite, d'influencer le résultat final en réglant quelques paramètres. Le but est aussi de fournir un outil simple afin qu'un animateur débutant obtienne rapidement un résultat, les réglages qu'il devra faire devront donc être les plus intuitifs possibles.

La représentation la plus répandue pour des formes 3D est la représentation surfacique, dans cet exposé il sera question de surfaces triangulées mais les méthodes utilisées sont transposables à d'autres types de surfaces. Par la suite nous cherchons donc un squelette de forme 3D capturant au mieux sa morphologie.

1.2 Etat de l'art

De nombreux algorithmes ont été proposés pour générer automatiquement des squelettes. Certains donnent des résultats non exploitables pour l'animation. Par exemple la construction de l'*Axe Médian* ([Blu67]) conduit à un squelette comportant des parties surfaciques, un post traitement est nécessaire afin de les éliminer. De plus l'*Axe Médian* est très sensible au bruit et demande une étape de "pruning" afin d'éliminer les nombreuse petites branches qui en résulte ([AST95]).

Parmi les techniques fournissant un squelette dont la structure est un graphe, on peut citer celles utilisant une voxelisation [WP02]. Elles demandent un temps de calcul important, et la taille des caractéristiques prises en compte dans le résultat dépend de la taille des voxels.

Dans [MWO03], un champ de forces répulsives est utilisé pour placer les nœuds du squelette. Les positions d'équilibre d'une particule chargée dans un champ électrique généré par une charge située à la surface du modèle donnent l'emplacement des nœuds. Cette méthode est lente et ne garantie pas que le résultat capturera toutes les caractéristiques du modèle.

Le graphe de Reeb repos sur des propriétés mathématiques prouvant qu'il capture dans une certaine mesure la morphologie des formes 3D, son intérêt est présenté dans [BMMP03]. Les nœuds du graphe de Reeb correspondent aux points critiques d'une fonction définie sur la surface étudiée. Plusieurs fonctions peuvent être utilisées, le choix dépend essentiellement de l'application visée. Le graphe de Reeb trouve de nombreuses applications ([TS04], [TDV06], [BRS03], [HSKK01]). Il donne des critères permettant de classer les modèles 3D, par exemple dans une base de données.

Dans [LV99], Lazarus introduit le *Level Set Diagram*, une nouvelle structure topologique, dont la construction est proche de celle du graphe de Reeb est qui permet de calculer un squelette d'un modèle 3D. Cependant la fonction utilisée possède de nombreux extrema locaux ce qui conduit à un graphe bruité lorsque le maillage des surfaces est très fin.

Dans [NGH04], la fonction utilisée est solution de l'équation de Laplace. Elle a la propriété de ne pas avoir de points critiques non désirés et permet donc de construire un graphe de Reeb non bruité quelque soit le modèle.

La première idée exploitée dans mon travail fut d'utiliser la fonction de [NGH04] pour construire un *Level Set Diagram* qui soit exploitable dans le sens où il serait stable, invariant et décrirait la morphologie de l'objet.

1.3 Contributions

Dans un premier temps mon travail a consisté à construire le *Level Set Diagram* en utilisant la fonction utilisée dans [NGH04]. Après analyse des résultats il est apparu que les squelettes obtenus possèdaient encore quelques défauts qui rendaient leur exploitation difficile. En effet des erreurs numériques introduisent malgré tout du bruit sur le squelette. D'autre part les squelettes ne rendent pas toujours compte de la symétrie du modèle, caractéristique importante de

la forme 3D. Je propose ici une méthode permettant de pallier à ces deux problèmes. Les squelettes obtenus permettent ainsi de mieux rendre compte de la morphologie des modèles ce qui, par exemple, rendrait leur classement dans une base de données plus pertinent.

De plus je propose une méthode inspirée de [LV99] permettant de plonger les squelettes pour que visuellement ils ressemblent à ceux créés par des animateurs professionnels. Dans le domaine de l'animation les bipèdes et les quadrupèdes représentent une grande part des modèles. Je propose également un plongement particulier des squelettes pour ces catégories de modèles afin d'obtenir des squelettes très proches des squelettes d'animation construits manuellement par des infographistes.

Chapitre 2

Calcul du graphe des Contours

2.1 Bases Mathématiques

La théorie de Morse rend compte du type d'homotopie d'une surface M à l'aide d'une fonction de Morse $f\colon M \to \mathbb{R}$ et de ses points critiques. Dans cette section je rappelle la définition d'une fonction de Morse pour les surfaces lisses puis je présente son adaptation aux fonctions linéaires par morceaux en définissant des points critiques malgré l'absence de dérivées. C'est cette théorie qui nous permettra de construire un graphe qui rendra compte de la morphologie des modèles et à partir duquel on construira un squelette significatif.
Cette section est largement inspirée de celle présentée dans [NGH04].
On rappelle qu'une 2-variété est une surface dont chaque point possède un voisinage homéomorphe à un disque.

2.1.1 Fonction de Morse sur une surface lisse

Milnor donne dans [Mil63] une description concise mais complète et surtout abordable de la théorie de Morse. Soit $p \in M \subset \mathbb{R}^3$ un point sur une surface (2-variété) M avec un voisinage continuement paramétrisé, avec (x_1, x_2) les coordonnées locales. Soit $f\colon M \to \mathbb{R}$ une fonction à valeurs réelles définie sur M. Voici quelques définitions et propriétés :

- Un point est *critique* si son gradient s'annule *i.e.* $\frac{\partial f}{\partial x_2} = \frac{\partial f}{\partial x_1} = 0$; autrement il est *régulier*.
- Un point critique est *dégénéré* si sa matrice hessienne $[\frac{\partial^2 f}{\partial x_i \partial x_j}]$ est singulière, autrement il est dit *de Morse*.
- Si tous les points critiques sont de Morse alors f est une appelée une *fonction de Morse*.
- L'*indice* d'un point critique de Morse est le nombre de valeurs propres négatives de sa matrice hessienne. Il indique le nombre de "directions descendantes" principales. Ces directions correspondent à des vecteurs propres.
- Un point critique de Morse sur une 2-variété est d'indice 0, 1 ou 2 soit respectivement un minimum, un point selle ou un maximum.

2.1.2 Fonction de Morse sur une surface triangulée

Banchoff [Ban67] a montré que la définition de fonction de Morse s'étend aux surfaces triangulées. Soit f une fonction à valeurs réelles linéaire par morceaux définie sur les sommets d'une 2-variété orientable triangulée M dont on étend les valeurs linéairement sur les arêtes et les faces du maillage. Pour le moment on suppose que pour toute arête $\langle v_1, v_2\rangle \in M$, $f(v_1) \neq f(v_2)$. On a donc la propriété suivante :

– le gradient de f est constant, non nul et parfaitement défini sur les faces, les points critiques de f se trouvent aux sommets.

On va essayer de caractériser et de classer ces points comme pour le cas lisse, pour cela on utilise la liste de définitions et de propriétés suivantes. Elles sont illustrées par la figure 2.1 :

– soient $Lk(v)$ le *lien* du sommet v, défini comme le graphe contenant les m sommets $v_1, v_2, ..., v_m$ qui partagent une arête avec v ainsi que les arêtes $\langle v_1, v_2\rangle, \langle v_2, v_3\rangle, ..., \langle v_m, v_1\rangle$,
– $Lk^+(v) = \{v_i \in Lk(v) \mid f(v_i) > f(v)\} \cup \{\langle v_i, v_j\rangle \in Lk(v) \mid f(v_i) > f(v) \text{ et } f(v_j) > f(v)\}$ le *lien supérieur*,
– $Lk^-(v) = \{v_i \in Lk(v) \mid f(v_i) < f(v)\} \cup \{\langle v_i, v_j\rangle \in Lk(v) \mid f(v_i) < f(v) \text{ et } f(v_j) < f(v)\}$ le *lien inférieur* et
– $Lk^\pm(v) = \{\langle v^+, v^-\rangle \mid f(v^-) < f(v) < f(v^+)\}$ les arêtes mixtes. $\#Lk^\pm(v)$ est toujours pair.
– $Lk(v) = Lk^+(v) \cup Lk^-(v) \cup Lk^\pm(v)$.

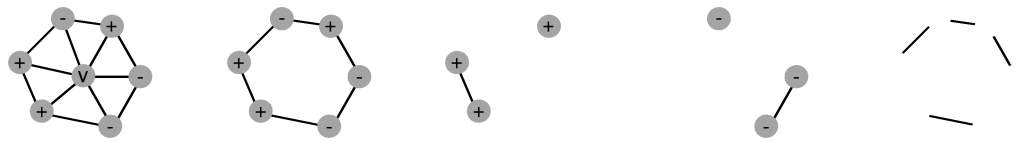


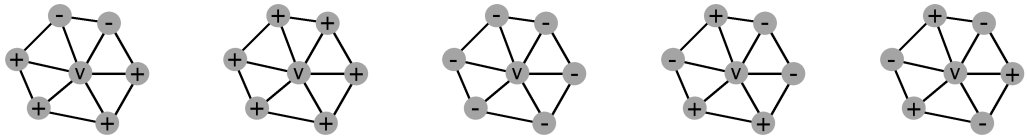
FIG. 2.1 – **(a)** : l'étoile de v , **(b)** : $Lk(v)$, **(c)** : $Lk^+(v)$, **(d)** : $Lk^-(v)$ et **(e)** : $Lk^\pm(v)$

On peut alors classer les points critiques et définir leur indice (analogue de la version lisse) comme suit (des exemples sont donnés sur la figure 2.2) :

$Lk^-(v) = \emptyset$	$\Rightarrow$	v est minimum et son indice vaut 0,
$Lk^+(v) = \emptyset$	$\Rightarrow$	v est maximum et son indice vaut 2,
$\#Lk^\pm(v) = 2$	$\Rightarrow$	v est régulier,
$\#Lk^\pm(v) = 2 + 2m, m \geq 1$	$\Rightarrow$	v est un point selle, d'indice 1 et de multiplicité m.

Si par convention on fixe la multiplicité des extrema à $m = 1$ alors Banchoff [Ban67] a montré que l'on peut calculer la caractéristique d'Euler ainsi :

$$\chi(M) = \sum_{v \in M, v \text{ critique}} (-m)^{indice(v)}$$



regulier $m = 0$ | *minimum* $ind = 0, m = 1$ | *maximum* $ind = 2, m = 1$ | *selle de Morse* $ind = 1, m = 1$ | *selle de singe* $ind = 1, m = 2$

FIG. 2.2 – Exemple de points régulier et critiques.

Dans la suite, par abus de langage, on appelera f une *fonction de Morse* comme dans le cas lisse.

2.1.3 Graphe de Reeb et graphe des contours

Nous disposons maintenant de fonctions de Morse qui vont nous permettre d'introduire dans le cas lisse le graphe de Reeb [Ree46] et dans le cas de surfaces triangulées le graphe des contours [vKvOB+97]. Ces graphes capturent la forme du modèle et on verra par la suite que leur plongement dans $\mathbb{R}^3$ permettra d'obtenir un squelette.
Dans ce qui suit, M désignera une surface orientable aussi bien lisse que triangulée.

Définition 1 (Ligne de niveau) *Soit f une fonction de Morse définie sur M. On appelle* ligne de niveau *correspondant à la valeur $a \in \mathbb{R}$, que l'on note $f^{-1}(a)$, l'ensemble des points x de M tels que $f(x) = a$.*

Propriété 1 *Soit f une fonction définie sur une surface M compacte (fermée et bornée), et $a \in \mathbb{R}$ une valeur régulière de $f : M \to \mathbb{R}$. Alors $f^{-1}(a)$ est une* 1-variété *sans bord (homéomorphe à un cercle) si M est sans bord et qui peut avoir des bords (homéomorphe à un demi-cercle) dans le cas contraire.*

$f^{-1}(a)$ est donc un ensemble de courbes reposant à la surface de M, en quelque sorte des "iso-contours" de M pour les valeurs de f.

Définition 2 (Graphe de Reeb) *Soit $f : M \to \mathbb{R}$ une fonction définie sur une surface M lisse et compacte. Le graphe de Reeb de f est l'espace quotient de M par la relation d'équivalence $\sim$ définie par :*

$$x_1 \sim x_2 \Leftrightarrow \begin{cases} f(x_1) = f(x_2) \\ x_1 \text{ et } x_2 \text{ sont dans la même composante connexe de } f^{-1}(f(x_1)) \end{cases}$$

Concrètement le graphe de Reeb et le graphe des contours sont composés de nœuds, correspondant aux points critiques de f, et d'arêtes correspondant aux composantes connexes reliant ces points critiques. La figure 2.3 montre le graphe de Reeb d'une fonction de hauteur sur un tore.

Pour le graphe des contours on reprendra la même définition que celle du graphe de Reeb mais avec M une surface triangulée. On commet là un abus de langage, le graphe considéré ici étant en fait une simplification du graphe des contours défini dans [vKvOB+97].

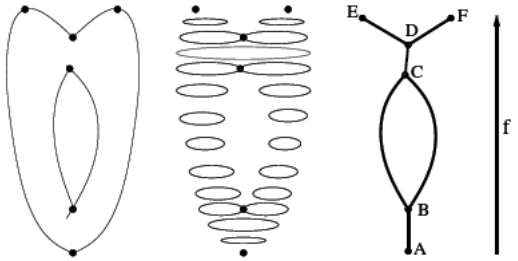


FIG. 2.3 – De gauche à droite : Points critiques de la fonction de hauteur f. Quelques lignes de niveau de f. Graphe de Reeb de f.

Dans [CMEH+03] est proposé un algorithme en $O(nlog(n))$ pour construire ce graphe, il sera décrit dans la section suivante. Il nécéssite une structure de données assez complexe et je me suis contenté d'en implémenter une version plus simple mais moins efficace lorsque l'on traite un point selle (il y a en général très peu de points selles par rapport au nombre total de sommets du maillage).

2.2 Construction du graphe

On travaille sur une surface triangulée M, on appelle S l'ensemble de ses sommets. On se donne les valeurs d'une fonction de Morse f définie sur S. On suppose que M est orientable et sans bord (il faudrait modifier un peu l'algorithme pour une surface avec des bords, voir [CMEH+03]). Les lignes de niveau correspondant à des valeurs régulières seront donc des courbes fermées.

Le principe de l'algorithme est simple, il faut balayer l'ensemble des points du maillage par ordre croissant de valeur de la fonction f. Voilà ce que l'on fait à chaque point v :

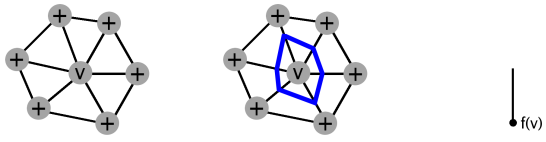
- on choisit $\varepsilon \in \mathbb{R}$ tel que : $\varepsilon < min\{|f(v) - f(v_i)|, / v_i \in Lk(v)\}$
- on fait évoluer $f^{-1}(f(v) - \varepsilon)$, déjà calculée, en $f^{-1}(f(v) + \varepsilon)$.
- si v est un point critique alors :
  - on rajoute un nœud correspondant à la valeur $f(v)$ au graphe
  - on déduit des changements topologiques entre $f^{-1}(f(v) - \varepsilon)$ et $f^{-1}(f(v) + \varepsilon)$ les arêtes à rajouter au graphe.

Cela est expliqué et illustré en détail par la suite, pour chaque type de point rencontré.

2.2.1 Détails de l'algorithme

Minimum

Lorsque v est un minimum alors $f^{-1}(f(v) + \varepsilon)$ contient une composante connexe de plus que $f^{-1}(f(v) - \varepsilon)$, ce qui se traduit par l'ajout d'un nœud et d'une demi-arête dans le graphe (on ne connait pas encore l'autre extrémité de cette arête), voir figure 2.4 :

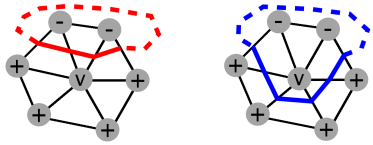


(a) (b) (c)

FIG. 2.4 – Ligne de niveau à proximité de v juste avant le passage de l'algorithme par v (a), et juste après (b), morceau rajouté au graphe après le passage par v (c).

Régulier

Lorsque v est régulier alors $f^{-1}(f(v)+\varepsilon)$ est homotope à $f^{-1}(f(v)-\varepsilon)$, le graphe n'est pas modifié, il faut juste mettre à jour la ligne de niveau comme indiqué sur la figure 2.5 :

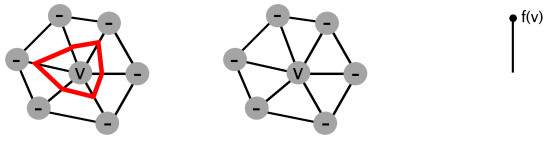


(a) (b)

FIG. 2.5 – Ligne de niveau à proximité de v juste avant le passage de l'algorithme par v (a), et juste après (b).

Maximum

Lorsque v est un maximum alors $f^{-1}(f(v)+\varepsilon)$ contient une composante connexe de moins que $f^{-1}(f(v)-\varepsilon)$, ce qui se traduit par l'ajout d'un nœud dans le graphe qui va devenir la deuxième extrémité d'un arête déjà existante, voir figure 2.6 :

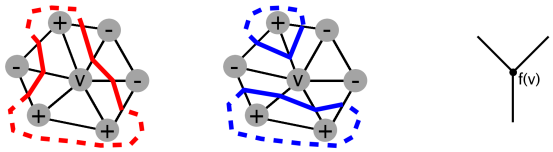


(a) (b) (c)

FIG. 2.6 – Ligne de niveau à proximité de v juste avant le passage de l'algorithme par v (a), et juste après (b), morceau rajouté au graphe après le passage par v (c).

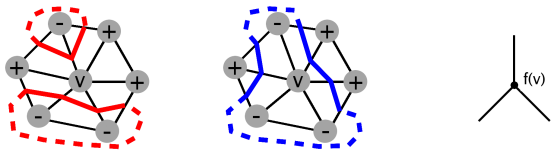
Selle

Lorsque v est un point selle, on ajoute un nœud et des arêtes au graphe, ces dernières reflètent les changements entre $f^{-1}(f(v)+\varepsilon)$ et $f^{-1}(f(v)-\varepsilon)$. *e.g.* sur les figures 2.7 et 2.8, une composante se divise en deux au passage par v donc dans le graphe il faut ajouter un point qui va être l'extrémité d'une arête déjà existante ainsi que deux nouvelles arêtes correspondant aux deux nouvelles composantes connexes de la ligne de niveau.



(a) (b) (c)

FIG. 2.7 – Ligne de niveau à proximité de v juste avant le passage de l'algorithme par v (a), et juste après (b), morceau rajouté au graphe après le passage par v (c).



(a) (b) (c)

FIG. 2.8 – Ligne de niveau à proximité de v juste avant le passage de l'algorithme par v (a), et juste après (b), morceau rajouté au graphe après le passage par v (c).

2.2.2 Implémentation et complexité

On représente une composante connexe de la ligne de niveau par une liste cyclique d'arêtes. Une ligne de niveau est une collection de ces listes. Pour avoir un algorithme efficace ces listes sont contenues dans des arbres de recherche équilibrés. Chaque insertion, suppression et recherche a donc une complexité logarithmique. Les opérations de fusion ou de division des composantes qui ont lieu aux points selles sont implémentées en concaténant ou divisant les arbres ce qui est aussi faisable en temps logarithmique.

Au final, chaque arête subit au plus une insertion, une suppression et 2 recherches et dans le pire des cas, à chaque sommet on effectue deux divisions et deux concaténations d'arbres (pour un point selle de multiplicité 1). Si on note n le nombre de triangles du maillage, comme il y a moins d'arêtes que de triangles alors chaque opération demande au plus $O(\log(n))$ donc au total, il faut au pire $O(n\log(n))$ pour construire le graphe.

2.3 Exemples de fonctions

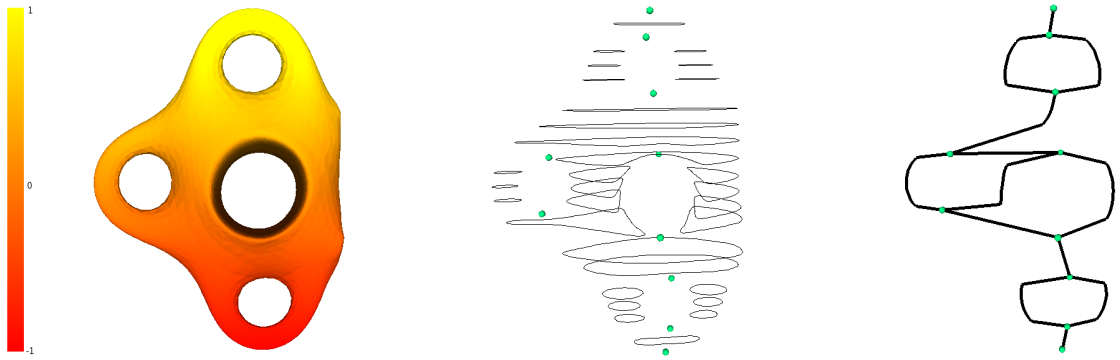
Le choix de la fonction de Morse dépend de l'application que l'on souhaite faire du graphe obtenu. Je présente ici quelques exemples de fonctions, d'autres ont été définies et utilisées dans [HSKK01],[TDV06].

2.3.1 Hauteur

Définition 3 (fonctions hauteur) *On munit l'espace d'un repère orthonormé* $(O,\vec{i},\vec{j},\vec{k})$. *Soit* M *une surface compacte et* $\vec{u} \in \mathbb{R}^3$ *un vecteur normé de l'espace.*
On définit alors $h_{\vec{u}} : M \to \mathbb{R}$ *la fonction hauteur associée à* $\vec{u}$ *telle que*

$$\forall P \in M \qquad h_{\vec{u}}(P) = \overrightarrow{OP} \,.\vec{u}$$

La figure 2.9 illustre les résultats obtenus avec une fonction de hauteur sur une surface de *genre 4*.



(a) (b) (c)

FIG. 2.9 – Valeurs de la fonction h sur le maillage (a), quelques lignes de niveau et les points critiques de h (b), son graphe des contours (c).

En pratique on prend souvent $\vec{i},\vec{j}$ ou $\vec{k}$ pour $\vec{u}$. Cependant, pour assurer l'invariance du graphe par rotation du modèle, on pourrait choisir la direction $\vec{u}$ comme la direction principale du modèle.

2.3.2 Distance

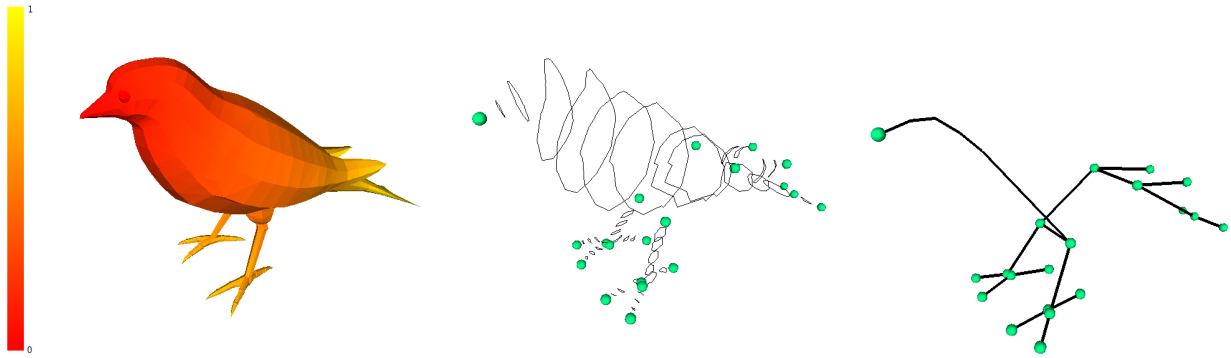
Définition 4 (fonctions distance à un point source) *Soit* M *une surface compacte. Soit une fonction* $dist : M^2 \to \mathbb{R}$ *qui associe une distance à chaque couple de points de la surface et soit* S *un point de la surface.*
On définit alors $d_{dist,S} : M \to \mathbb{R}$ *la fonction distance à un point source telle que*

$$\forall P \in M \qquad d_{dist,S}(P) = dist(S,P)$$

En pratique, pour un maillage, on travaille sur les sommets et on prend comme fonction *dist* la distance au sens de Dijkstra sur le graphe sommets-arêtes du maillage, chaque arête étant pondérée par sa longueur Euclidienne (voir [LV99]). On pourrait aussi calculer des distances géodésiques mais cela demanderait un temps de calcul élevé. Pour que le choix du point source soit déterministe, on peut choisir :

- le point le plus éloigné (au sens de Dijkstra) du point le plus proche (au sens de la distance euclidienne dans $\mathbb{R}^3$) du centre de gravité du modèle,
- le point le plus éloigné (au sens de Dijkstra) du point trouvé par la méthode précédente
- ou encore tout autre point dont les caractéristiques ne dépendent que du modèle et sont invariantes par transformation de celui-ci.

La figure 2.10 illustre les résultats avec le point source au bout du bec de l'oiseau.



(a) (b) (c)

FIG. 2.10 – Valeurs de la fonction d sur le maillage (a), quelques lignes de niveau de et les points critiques de d (b), son graphe des contours (c).

2.3.3 Fonction harmonique

Définitions et propriétés

Définition 5 (Equation de Laplace) *Soit M une surface compacte. Une fonction $T : M \to \mathbb{R}$ vérifie l'équation de Laplace sur M si et seulement si :*

$$\triangle T(P) = 0 \qquad \forall P \in M$$

Propriété 2 *Une fonction T qui satisfait l'équation de Laplace est dite harmonique. Elle a les propriétés suivantes :*

– *la valeur moyenne sur un petit disque est égale à la valeur au centre du disque (Propriété de la moyenne).*
– *elle ne possède donc pas d'extrema locaux hors de la frontière de M.*

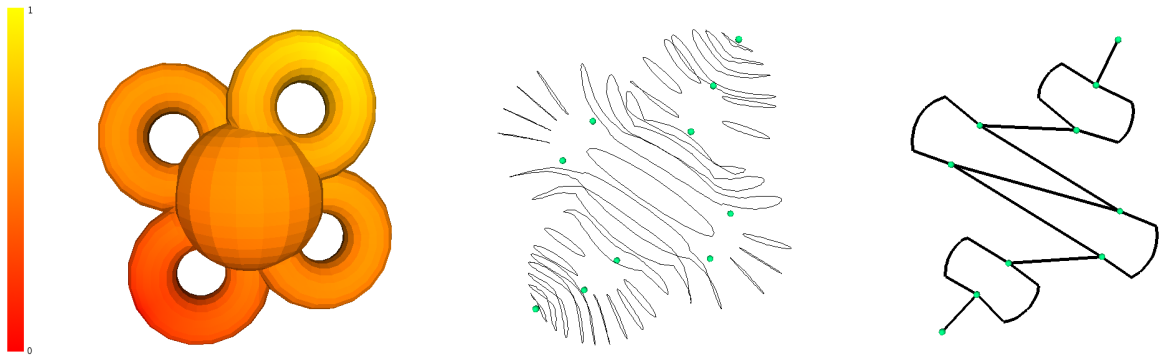
Définition 6 (Equation de Laplace avec conditions de Dirichlet non-homogènes) *Soit M une surface compacte,* Γ *sa frontière et g définie sur* Γ*. On définit le système suivant :*

$$\begin{cases} \triangle T(P) = 0 & \forall P \in M \\ T(P) = g(P) & \forall P \in \Gamma \end{cases} \quad (1)$$

Physiquement cela revient à considérer M comme un matériau homogène auquel on applique des contraintes de chaleur sur sa frontière. La solution correspond à la température du matériau en régime stationnaire.

Propriété 3 *Le système* (1) *admet une solution unique. On définit alors* $T_{g,\Gamma} : M \to \mathbb{R}$ *la fonction solution de ce système.*

La figure 2.11 montre la solution obtenue en fixant un maximum et un minimum manuellement. Aucun autre extremum n'est apparu.



(a) (b) (c)

FIG. 2.11 – Valeurs de la fonction T sur le maillage (a), quelques lignes de niveau et les points critiques de T (b), son graphe des contours (c).

Résolution de l'équation

Il faut résoudre trois problèmes :
– quels points choisir pour la frontière Γ ?
– quelles valeurs donner à g en ces points ?
– résoudre numériquement le système.

Il peut être intéressant de laisser à l'utilisateur le choix des points extrema, afin qu'il puisse décider lui même des caractéristiques qu'il souhaite voir transparaître du modèle vers le graphe. Par exemple sur un modèle de cheval, il peut souhaiter animer les oreilles mais pas la queue, il

peut alors spécifier un extremum sur chaque partie qu'il souhaite animer indépendemment du reste du corps, *e.g.* les oreilles mais pas la queue. Il peut aussi être intéressant que l'utilisateur n'ait rien à faire ou qu'au moins il soit guidé dans le choix des points caractéristiques qu'il pourra ensuite conserver ou éliminer. Deux méthodes d'extraction de ces points sont décrites ci-après.

Les techniques présentées dans [TDV06] et dans [SDG05] reposent sur le choix d'un point source puis l'extraction des points se trouvant à des extrema locaux en terme de distance à ce point source. [TDV06] calcule les distances avec l'algorithme de Dijkstra et [SDG05] résoud l'équation de Poisson ($\triangle T = - \parallel x \parallel$) ce qui donne une fonction T qui présente aussi des extrema locaux aux points les plus éloignés localement. Ces deux techniques nécessitent ensuite un regroupement des points trop proches (clusterisation). En effet dans des zones en bout de branches et assez "plates" on trouvera beaucoup de points extrema. La figure 2.12 extraite de [SDG05] illustre le problème. Il faut donc que l'utilisateur fixe un paramètre qui est la taille des clusters utilisés pour extraire un seul extremum par cluster (logiquement choisi comme celui ayant la valeur la plus grande et étant le plus éloigné).

(a) (b) (c)

FIG. 2.12 – Equipotentielles de la solution de l'équation de Poisson avec condition de Dirichlet homogène sur le museau (a), extremum avant clusterisation (b), extremum après clusterisation (c).

Pour les valeurs à donner à g sur les points de Γ, on peut :
- fixer $g(S) = 0$ pour le point source et $g(P) = 1$ pour les autres points $P \in \Gamma$
- ou prendre la valeur obtenue lors de l'extraction des points de Γ, c'est-à-dire que plus les points sont éloignés du point source plus ils auront une grande valeur.

La seconde méthode donnera pour la suite de meilleurs résultats car elle donne un poids moins important aux branches moins longues, ce qui évite d'avoir un gradient trop élevé sur celles-ci.

Pour ce qui est de la résolution numérique du système (1), j'ai utilisé une méthode par éléments finis de type $\mathbb{P}_1$ (triangles, fonctions linéaires). Ces éléments correspondent parfaitement aux besoins du problème car les modèles sont triangulés et la fonction calculée est définie aux sommets du maillage puis interpolée linéairement sur le reste de la surface. La qualité du résultat dépend du maillage, ainsi des triangles avec un angle trop aigü ou une aire trop grande par rapport au reste des triangles peuvent conduire à des inexactitudes dans le résultat. La matrice assemblée étant très creuse, on peut l'inverser en utilisant le solveur SuperLU, on obtient ainsi des temps de calcul raisonnables même sur de gros maillages (voir tableau 2.1).

Modèle	nb. sommets	dim. matrice	termes non nuls	tps de calcul
dromadaire	2667	7.11×10^6	0.260%	$0s061ms$
cheval	48485	2.35×10^6	0.014%	$4s050ms$
girafe	68844	4.74×10^6	0.010%	$5s439ms$
lionne	95138	9.05×10^6	0.007%	$9s752ms$

TAB. 2.1 – Temps de calcul sur différents modèles et mise en évidence des faibles taux de termes non nuls dans la matrice.

2.4 Choix de la fonction la plus adaptée à l'animation

La meilleure fonction est dans ce contexte celle qui peut capturer au mieux les caractéristiques géométriques et morphologiques des modèles. Il faut donc écarter les fonctions de hauteur qui ne prennent en compte que les caractéristiques dans une direction donnée comme on peut le voir sur la figure 2.13.

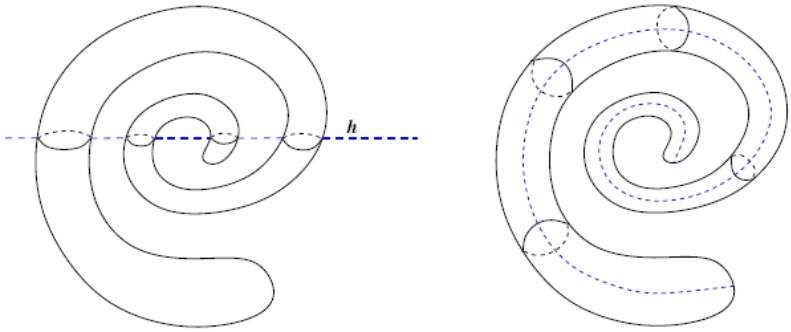
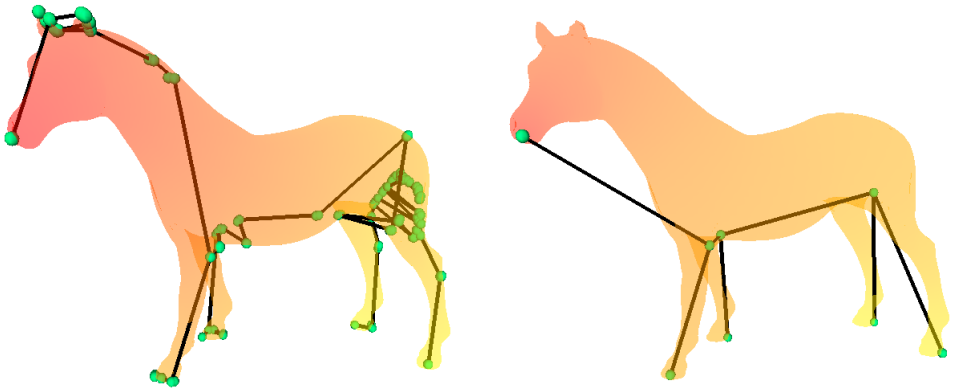


FIG. 2.13 – Exemple d'objet pour lequel une fonction de hauteur ne peut pas donner de résultat satisfaisant (à gauche). On obtient un squelette adapté avec une fonction de distance ou une fonction harmonique en prenant un point source à une extrémité de l'objet (à droite)

Si les autres fonctions permettent de capturer les caractéristiques des objets, la fonction de distance avec Dijkstra donne un résultat très bruité lorsque les modèles sont très fortement maillés ou bruités. D'après la *propriété 2* une fonction harmonique ne possèdera pas de points critiques superflus, exception faite d'erreurs numériques qui restent rares. La figure 2.14 permet de comparer ces deux fonctions sur un modèle très maillé. Le résultat est optimal avec la fonction harmonique en ayant fixé un extremum à chaque patte et un sur le museau.

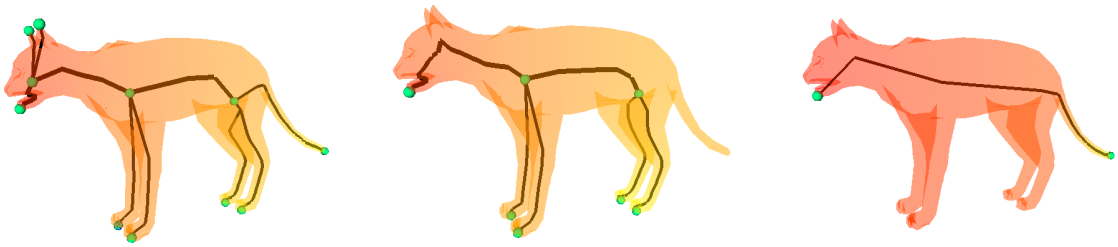
Le contrôle donné à l'utilisateur dans le choix des points sources est intéressant pour l'animation car il lui permet de gérer l'aspect du squelette de base obtenu. Ce contrôle est illustré sur la figure 2.15 sur l'exemple du chat.

Les autres fonctions connues que nous avons testées ne donnent pas de meilleurs résultats pour l'application visée. Tous ces critères font que la fonction harmonique semble être la meilleure pour faire un squelette d'animation.



(a) (b)

FIG. 2.14 – (a) Graphe des contours pour la fonction distance : 92 nœuds, (b) Graphe des contours pour la fonction harmonique : 8 nœuds.



(a) (b) (c)

FIG. 2.15 – Le museau, les oreilles, les pattes et la queue ont été sélectionnés (a), seuls le museau et les pattes sont sélectionnés (b), un maximum et un minimum suffisent en théorie (c).

Chapitre 3

Simplification du graphe

3.1 Nécessité de simplifications

Il arrive que même en utilisant une fonction harmonique on obtienne quelques défauts sur le graphe des contours, à cause d'erreurs numériques. Cela se remarque par des branches de l'arbre qui ne correspondent pas à des extrema choisis : il y a des arêtes $\langle extremum - selle \rangle$ indésirables. Le graphe présente aussi, et plus souvent, une dissymétrie alors que le modèle lui est symétrique. Or notre but est de capturer les caractéristiques du modèles, il a donc fallu chercher un moyen d'éliminer ces défauts qui correspondent à des arêtes $\langle selle - selle \rangle$.

3.2 Poids d'une arête

On associe une valeur (ici, les valeurs de la fonction de Morse f) aux nœuds d'un graphe. Le *poids* d'une arête reliant deux points critiques du graphe est définie comme la valeur absolue de la différence des valeurs en ces nœuds. Le poids permet de donner une priorité aux arêtes lors du processus de simplification. La simplification des parties inutiles du graphe se fait par ordre de poids croissant, ce qui convient particulièrement aux problèmes énoncés plus haut car les arêtes concernées ont un poids en général très faible.
En pratique on calculera pour chaque arête son poids :

$$\delta(\langle v_1, v_2 \rangle) = |f(v_1) - f(v_2)|$$

puis en utilisant le diamètre en terme de valeur de fonction de Morse du graphe :

$$diam(G) = |\max_{v_i \in G}(f(v_i)) - \min_{v_i \in G}(f(v_i))|$$

on calcule le poids normalisé de chaque arête :

$$\overline{\delta}(\langle v_1, v_2 \rangle) = \frac{\delta(\langle v_1, v_2 \rangle)}{diam(G)}$$

On a ainsi $0 \leq \overline{\delta} \leq 1$, qui nous renseigne sur le poids de chaque arête dans le graphe.

3.3 Filtrage selon le poids

3.3.1 Arêtes Extremum-Selle

Si le modèle est très bruité, des erreurs numériques peuvent introduire des extrema non désirés. On peut donc dans ce cas essayer de les supprimer. On fixe alors un paramètre $0 \leq \alpha \leq 1$ et on supprime toutes les arêtes $\langle extremum - selle \rangle$ du graphe qui ont un poids inférieure à α. Il est important de supprimer les arêtes dans l'ordre croissant de leur poids, afin de ne pas supprimer des détails qui ne devraient pas l'être (voir figure 3.1). L'algorithme ci-dessous permet d'effectuer ces simplifications.

Le choix de α peut être laissé à l'utilisateur afin de régler la finesse du squelette. Par exemple, si la détection des extrema a repéré les pattes, la queue et les oreilles, alors l'utilisateur, en choisissant un α assez grand pourra éliminer les oreilles du squelette, celles-ci ayant normalement un poids plus faible que les pattes pour la fonction de Morse choisie.

Cependant si le but est d'éliminer le bruit du squelette alors une détection automatique du α correct est possible. En effet, comme on peut le voir sur la figure 3.2, le signal bruité est nettement distinguable du reste, le poids des arêtes inutiles étant très faible par rapport à celle des autres. De plus dans le cas où l'on a choisi une fonction harmonique pour f, on pourrait ne garder que les arêtes correspondant aux extrema choisis. L'avantage du filtrage par poids est qu'il fonctionne avec toutes les fonctions.

Algorithm 1 *Filtrage*

- Initialisation -
L : une liste de priorité triée par poids croissant
for all a arête extrema-selle de G **do**
  if $\overline{\delta}(a) \leq \alpha$ **then**
    ajouter a dans L
  end if
end for

- Main Loop -
while L non vide **do**
  extraire $a =$ premier élément de L
  supprimer a de G
  if une arête $b \neq a$ est supprimée de G **then**
    supprimer b de L si nécessaire
  end if
  if une nouvelle arête extrema-selle c apparait **then**
    if $\overline{\delta}(c) \leq \alpha$ **then**
      ajouter c dans L
    end if
  end if
end while

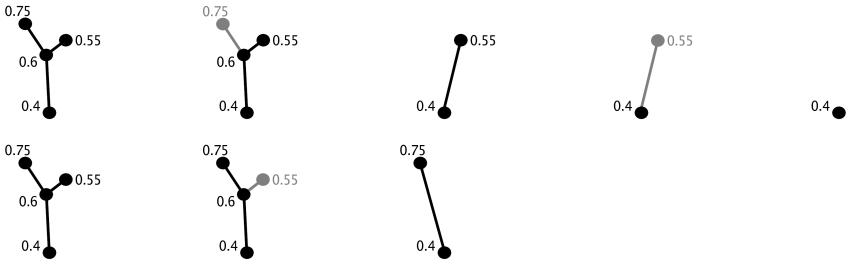
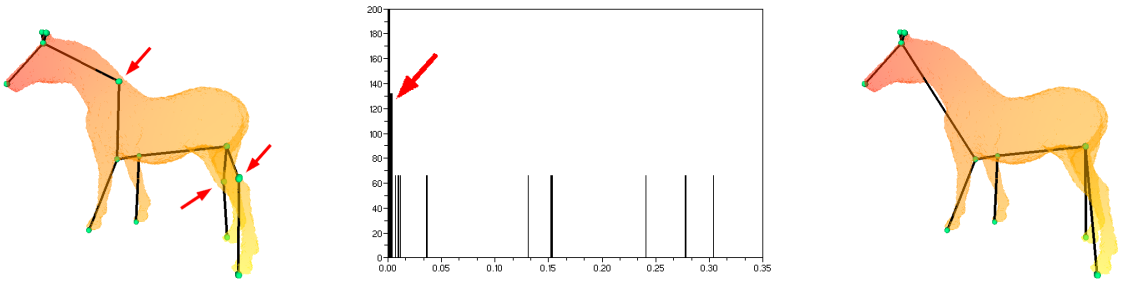


FIG. 3.1 – En haut : étapes successives de l'algorithme avec $\alpha = 0.18$ et sans trier par poids croissant : le squelette se réduit à un nœud. En bas la même chose mais en triant : le résultat est celui souhaité.



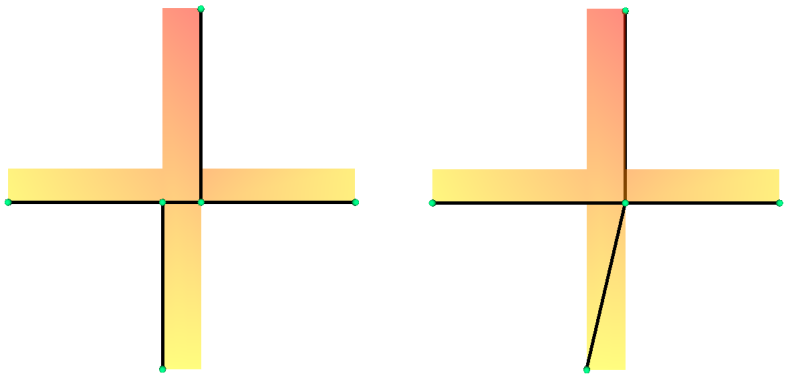
(a) **(b)** **(c)**

FIG. 3.2 – Graphe obtenu sur un modèle très bruité, le bruit est mis en évidence par les flèches **(a)**, le spectre de le poids des arêtes **(b)**, le graphe obtenu après filtrage avec $\alpha = 0.02$ **(c)**.

3.3.2 Arêtes Selle-Selle

Par construction, un modèle même parfaitement symétrique ne donnera pas un graphe symétrique, or nous cherchons à avoir un graphe qui reflète (parfaitement) la forme du modèle et donc ses symétries. En prenant l'exemple de la figure 3.3, on voit que l'on obtient deux points selles de multiplicité 1 alors que l'on voudrait un seul point selle de multiplicité 2. Or un nœud de multiplicité 2 ne peut être engendré que par un sommet dont la valence dans le graphe est au moins égale à 6. Donc même dans un cas idéal où la fonction de Morse serait parfaitement symétrique, un modèle ne possédant pas un tel sommet au bon endroit ne donnerait pas un graphe symétrique.

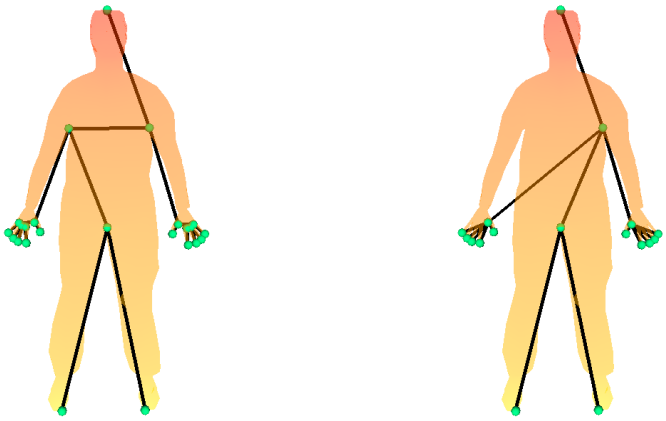
Il est cependant possible de corriger le graphe obtenu en effectuant un filtrage similaire à celui utilisé pour éliminer le bruit mais qui concerne cette fois les arêtes $\langle selle, selle \rangle$. En effet une arête qui relie deux points critiques symétriques l'un par rapport à l'autre aura un poids très faible car les fonctions harmoniques (ou distances) possèdent la même symétrie que le modèle si les points sources sont choisis en fonction de cette symétrie. En fixant un paramètre $0 \leq \beta \leq 1$ et en contractant toutes les arêtes a_i reliant deux points selle et telles que $\overline{\delta}(a_i) \leq \beta$ on peut alors donner au graphe la symétrie du modèle. Des exemples sont donnés sur les figures 3.3, 3.4 et 3.5.



(a) **(b)**

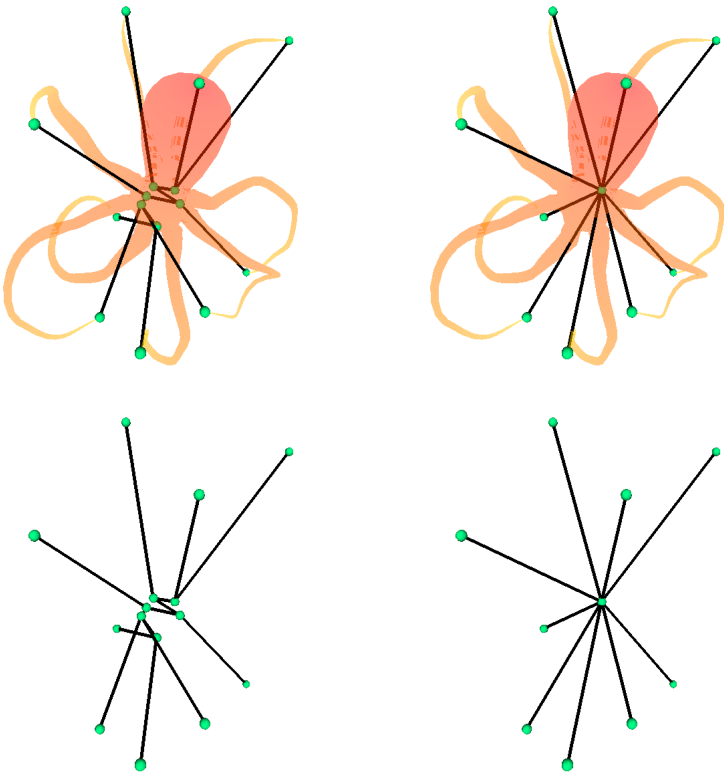
FIG. 3.3 – Graphe non symétrique obtenu à partir d'un modèle présentant une symétrie **(a)**, le même graphe obtenu en supprimant l'arête de faible poids **(b)**.

Cette méthode permet de capturer des symétries complexes comme pour la pieuvre de la figure 3.5 bien que le modèle soit déformé. En effet la pieuvre ne possède pas réellement de symétrie car ses tentacules ne sont pas droites mais ce qui est important est que leurs huit bases soient régulièrement disposées par rapport au corps de l'animal.



(a) **(b)**

FIG. 3.4 – Graphe non symétrique obtenu à partir d'un modèle présentant une symétrie **(a)**, le même graphe obtenu en filtrant avec $\beta = 0.03$ **(b)**.

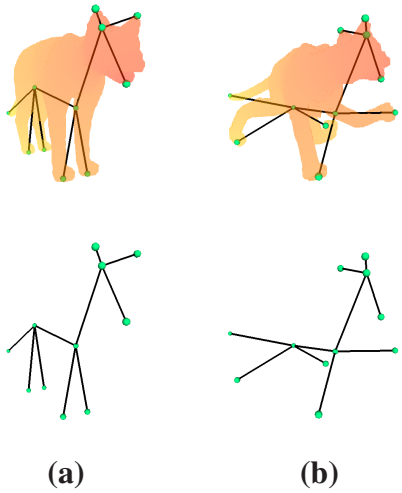


(a) **(b)**

FIG. 3.5 – Graphe non symétrique obtenu à partir d'un modèle présentant une symétrie axiale **(a)**, le même graphe obtenu en filtrant avec $\beta = 0.007$, le nœud obtenu a une valence de 9 **(b)**. Le graphe est simplifié au niveau des épaules et des mains.

3.4 Commentaires des résultats

Ce post-traitement sur le graphe des contours obtenu permet d'obtenir un graphe représentant mieux la morphologie du modèle. De plus ces simplifications assurent une plus grande stabilité du squelette lorsque l'objet subit des transformations affines mais aussi après animation de l'objet : le graphe reste topologiquement le même (voir figure 3.6). Cela peut être très utile dans beaucoup d'applications (*e.g.* le *shape-matching*) qui jusque là utilisent les graphes sans les simplifications décrites dans cette section.



(a) **(b)**

FIG. 3.6 – Le graphe simplifié obtenu pour le chat dans une position au repos **(a)**, le graphe simplifié obtenu pour le chat après déformation du modèle **(b)**. Ils sont topologiquement identiques.

Une autre application de ces simplifications pourrait être la construction de graphes multi-résolutions, le fait d'augmenter α supprimant les détails les plus petits, on peut obtenir plusieurs niveaux de détails comme on peut le voir sur la figure 3.7.

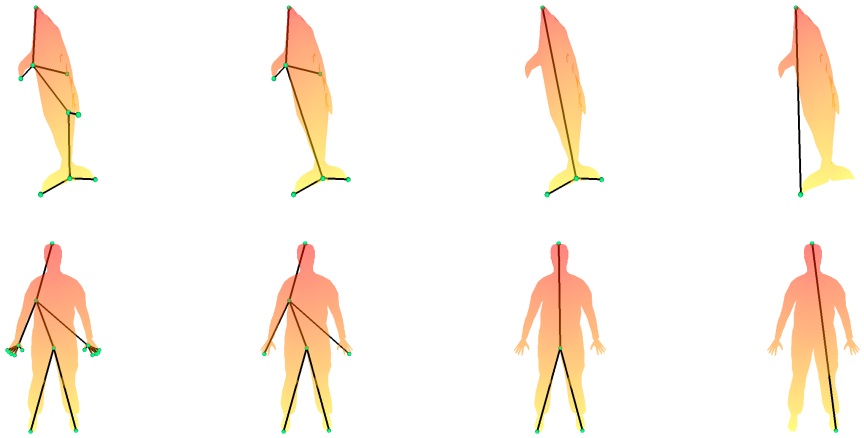


FIG. 3.7 – Différents niveaux de détails obtenus en augmentant la valeur de α.

Chapitre 4

Plongement du graphe : vers un squelette significatif

Dans le but d'obtenir un squelette, il faut plonger les nœuds du graphe des contours dans $\mathbb{R}^3$. [LV99] propose une méthode pour plonger un graphe des contours et obtenir ce qu'il appelle le *Level Set Diagram (LSD)*. Je me suis inspiré de cette méthode afin de plonger entièrement le graphe dans le but de se rapprocher d'un squelette visuellement proche d'un squelette d'animation.

4.1 Plongement simple

Soit a une valeur régulière d'une fonction de Morse et sa ligne de niveau $f^{-1}(a)$. Soit une composant connexe C de $f^{-1}(a)$. C est une courbe fermée composée de segments de droite dont les extrémités sont les points $(p_1, p_2, ..., p_k)$. On définit alors le centre de C, noté $centre(C)$ comme le barycentre de ses segments. La formule suivante permet de calculer $centre(C)$, en posant $p_{k+1} = p_1$:

$$centre(C) = \frac{\sum_{j=1}^{k} \|p_j p_{j+1}\| \frac{p_j + p_{j+1}}{2}}{\sum_{j=1}^{k} \|p_j p_{j+1}\|}$$

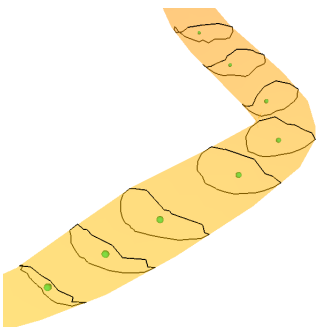


FIG. 4.1 – Quelques contours et leurs centres.

On choisit le barycentre des segments plutôt que le barycentre des points car le résultat est indépendant du niveau de discrétisation de la surface et que l'on évite ainsi les artéfacts donnés pour des maillages non uniformes.

Si un nœud du graphe des contours correspond à une valeur a régulière et à un contour C, on peut donc le plonger là où se trouve $centre(C)$. En général $centre(C)$ est à l'intérieur du modèle mais ce n'est pas forcément le cas si le contour est fortement non convexe. Actuellement le graphe des contours ne possède que des nœuds associés à des valeurs critiques. Ne sachant pas plonger ceux-ci on va faire intervenir de nouveaux nœuds que l'on sait plonger. Je décris dans la suite les transformations à faire subir au graphe.

Plonger un extremum

Soit v un sommet extremum et V le nœud qui le représente dans le graphe et soit $A = \langle V, U \rangle$ l'unique arête du graphe qui possède V comme extrémité. On suppose que v est un minimum, le cas du maximum est similaire. Soit une valeur proche de $f(v)$, notée $f(v)+\varepsilon$, on va plonger V au centre de la composante connexe correspondant à l'arête A du graphe et à la ligne de niveau $f^{-1}(f(v)+\varepsilon)$ comme sur la figure 4.2.

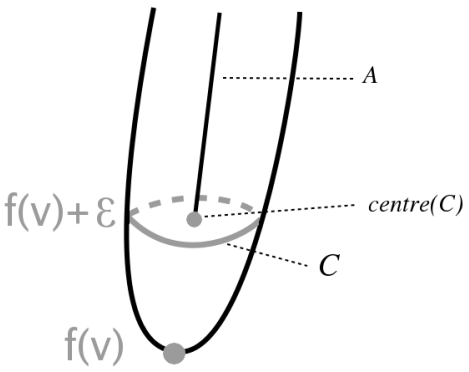


FIG. 4.2 – Plongement du point V correspondant à un minimum.

En pratique on peut choisir ε de deux façons différentes :
– ε tel que : $\varepsilon < min\{|f(v) - f(v_i)|, \ / \ v_i \in Lk(v)\}$
– ou $\varepsilon = \sigma |f(v) - f(u)|$ avec σ proche de 0, par exemple $\sigma = 5\%$

La première méthode permet de trouver un ε très proche de 0 et la seconde prend en compte le poids de l'arête A. Dans le cas des extrema j'ai choisi la seconde méthode qui permet d'éviter des artéfacts que l'on peut rencontrer à proximité des extrema : les contours y reflètent moins la forme de l'objet. La figure 4.3 illustre le plongement des extrema situés aux bouts des doigts d'une main humaine.

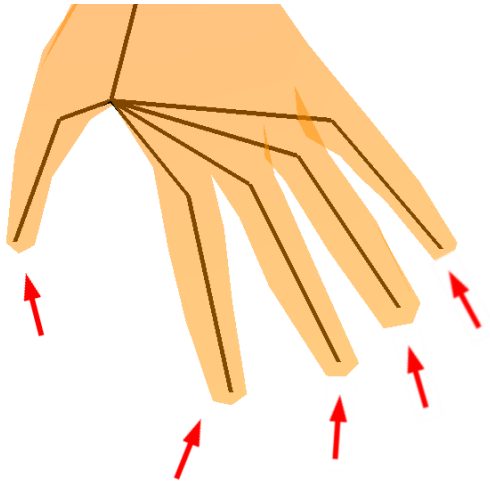


FIG. 4.3 – Plongement de cinq maxima, en prenant $\sigma = 5\%$.

Plonger un point selle

Soit v un sommet selle et V le nœud qui le représente dans le graphe et soit $A_i = \langle V, U_i \rangle$ les arêtes du graphe qui possèdent V comme extrémité. On va insérer un nœud sur chaque A_i :

- si $f(u_i) < f(v)$ alors on insère un nœud N_i qui correspond à un valeur $f(v) - \varepsilon$.
- si $f(u_i) > f(v)$ alors on insère un nœud N_i qui correspond à un valeur $f(v) + \varepsilon$.

On obtient ainsi $A_{i,0} = \langle V, N_i \rangle$ et $A_{i,1} = \langle N_i, U_i \rangle$ deux nouvelles arêtes qui remplacent A_i. Les nœuds N_i correspondent à des valeurs régulières et peuvent être plongés dans $\mathbb{R}^3$. La figure 4.4 illustre l'insertion de ces nœuds plongés.

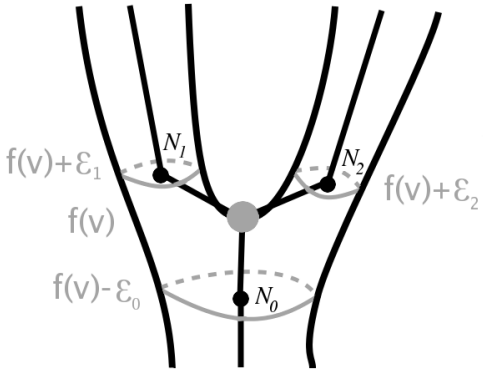


FIG. 4.4 – Insertion des points N_i pour le point selle v.

Il reste maintenant à plonger le nœud V. Pour cela on utilise l'isobarycentre des points N_i qui sera le plongement de V (voir figure 4.5). Après cette étape, on obtient un graphe complètement plongé dans R^3 qui se rapproche de plus en plus d'un squelette d'animation, quelques exemples sont présentés sur la figure 4.6. Dans le paragraphe suivant je montre comment améliorer ces résultats.

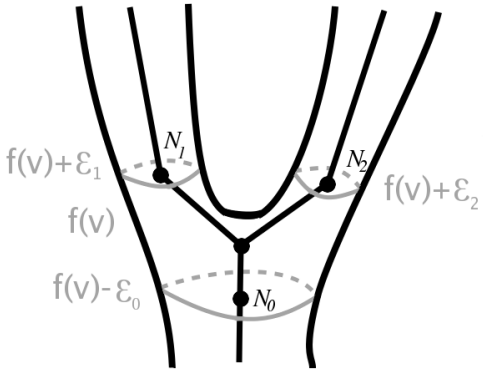


FIG. 4.5 – Insertion des points N_i pour le point selle v.

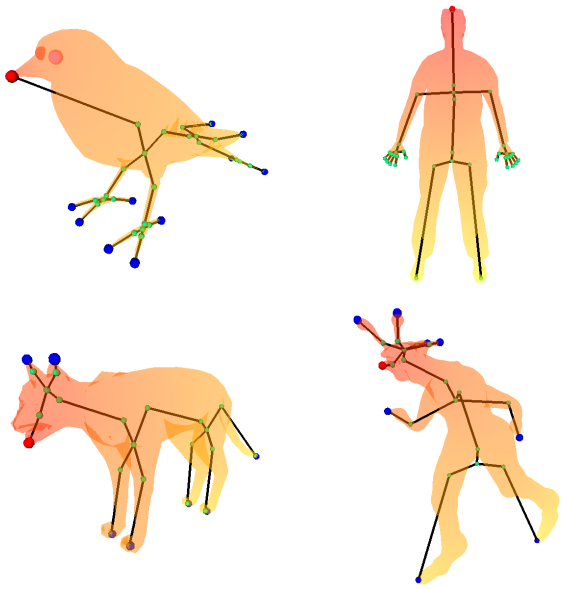


FIG. 4.6 – Différents squelettes plongés selon la méthode décrite précédemment.

Une chose importante que l'on remarque sur les squelettes obtenus est qu'ils respectent la symétrie du modèle lorsque celle-ci existe, la figure 4.7 illustre cette propriété sur l'exemple du chat.

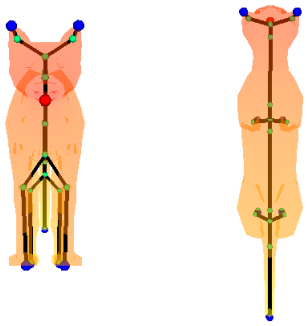


FIG. 4.7 – Mise en évidence de la symétrie du squelette.

4.2 Amélioration du plongement : prise en compte des symétries

En voyant les résultats précédents, figure 4.6, on remarque que l'on a introduit des de nœuds non significatifs dans les squelettes, ils sont mis en évidence sur la figure 4.8. Ces nœuds sont caractérisés par le fait qu'ils sont situés sur un axe de symétrie du graphe. On parle ici de symétrie au sens topologique pour le graphe, ce qui est plus important qu'une symétrie du plongement du graphe dans $\mathbb{R}^3$ car un modèle ayant un bras levé et l'autre baissé perd sa symétrie alors que son graphe la conserve. Dans un but de simplification du squelette, il faut donc être capable de détecter cette symétrie et d'en déduire les simplifications à faire, c'est ce que j'explique dans la suite.
A partir de maintenant je considère que les graphes ne comportent pas de cycles, dans le cas contraire, il faudrait des algorithmes un peu plus complexes.

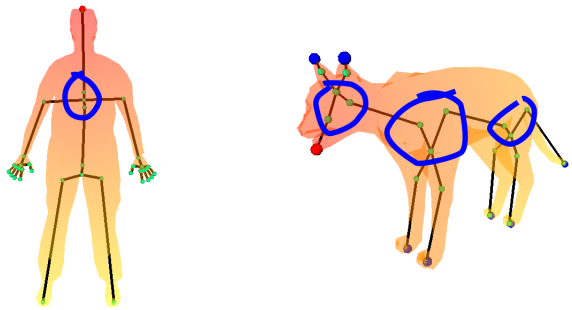


FIG. 4.8 – Mise en évidence de zones comportant des nœuds non significatifs.

Détection d'une symétrie principale

La symétrie que l'on cherche est un automorphisme de graphe qui laisse invariant un ensemble connexe d'arêtes du graphe. Leur recherche sur un graphe quelquonque et sans indication est un problème très difficile et très coûteux en temps de calcul. On se contentera donc d'une heuristique qui ne garantira pas le résultat mais permettra d'en trouver un convenable dans la plupart des cas. On impose à l'utilisateur de choisir le point source s sur l'axe de symétrie (*e.g.* la tête du personnage) ce qui va nous donner une première arête A_0 se trouvant sur l'axe de symétrie du graphe. Etant donné que le graphe ne possède pas de cycle, il peut être considéré comme un arbre dont la racine est le nœud S_0 correspondant au point source s.

On note $A_i = \langle S_i, S_{i+1} \rangle$ l'ensemble des arêtes se trouvant sur l'axe de symétrie principale. L'algorithme suivant décrit comment trouver A_{i+1}, si elle existe, connaissant A_i.

Algorithm 2 *Detection de symetrie*

Require: $A_0 = \langle S_0, S_1 \rangle$
Ensure: marque toutes les arêtes se trouvant sur l'axe de symétrie

$A = \langle S, S' \rangle \leftarrow A_0 = \langle S_0, S_1 \rangle$

while $A \neq \emptyset$ **do**

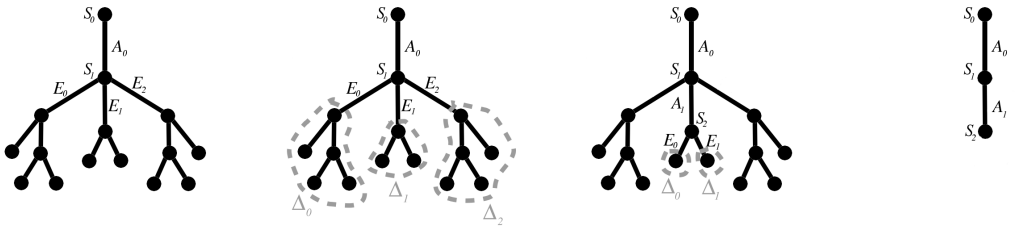
marquer A comme appartenant à l'axe de symétrie

$\{E_0, E_1, ..., E_k\}$ est l'ensemble des arêtes $E_j = \langle S', P_j \rangle$ adjacentes à S' et telles que $E_j \neq A$

$\Delta_j \leftarrow$ l'arbre dont P_j est la racine, privé du sous arbre contenant E_j.

$U_l \leftarrow$ ensemble d'arbres contenant des Δ_j tous égaux selon une heuristique.

if $\exists!\ U_l$ tel que $\#U_l = 1$ **then**
$A \leftarrow E_j \in U_l$ (U_l possède un unique élément).
else
$A \leftarrow \emptyset$
end if
end while



(a) **(b)** **(c)** **(d)**

FIG. 4.9 – **(a)** : initialisation, **(b)** création des Δ_j, Δ_0 et Δ_2 sont égaux, donc Δ_1 est le seul candidat : E_1 est sur l'axe de symétrie. **(c)** Deuxième itération, Δ_0 et Δ_1 sont égaux, il n'y a pas de candidat donc l'algorithme s'arrête. **(d)** L'axe de symétrie détecté.

Quelques remarques sur l'algorithme :

– L'algorithme s'arrête lorsque l'on n'obtient pas un unique ensemble U_l de cardinal égal à 1. En effet si U_l n'a qu'un élément alors celui-ci ne peut pas avoir d'image par la symétrie, à part lui même : il est sur l'axe de symétrie. Et s'il y a plusieurs ou aucun ensemble de cardinal égal à 1, on ne peut alors pas trouver une nouvelle arête sur l'axe donc l'algorithme s'arrête.

– Le fait de ranger les sous arbres Δ_i dans les ensembles U_l se justifie par le fait que si deux arbres sont topologiquement identiques alors il existe un isomorphisme évident entre ces deux arbres. Deux tels arbres peuvent donc être candidats pour être images l'un de

l'autre par la symétrie recherchée. Or prouver l'égalité topologique de deux arbres est très complexe. On choisit donc une heuristique : deux arbres sont dans le même ensemble U_l s'ils ont la même profondeur ou des racines de degrés égaux, etc. Ces critères peuvent être combinés pour affiner le test d'égalité.

- Il est intéressant de noter que les ensembles U_l peuvent contenir plus de 2 éléments, dans ce cas on détecte non plus simplement une symétrie axiale (invariance par rotation d'angle π) mais une invariance par rotation d'angle $\frac{2\pi}{\#U_l}$. C'est le cas pour la pieuvre : on obtient un ensemble $\#U_0 = nombre\ de\ tentacules = 8$ qui donne ce que l'on peut appeler une invariance du graphe par rotation d'angle $\frac{\pi}{4}$ et d'axe l'arête du graphe qui correspond au corps de l'animal.
- Cet algorithme est utilisable pour trouver des symétries non principales, en l'exécutant sur les sous-parties du graphe qui ne sont pas sur l'axe principal. On peut ainsi obtenir une hiérarchie de symétries comme dans [PSS06].

Simplification du plongement

Maintenant que l'axe de symétrie est connu on peut en déduire une simplification du plongement précédemment obtenu. On va pour cela fusionner les deux nœuds se trouvant sur l'axe de symétrie et de part et d'autre d'un point selle lui-même sur l'axe de symétrie.

Pour chaque point selle V se trouvant sur l'axe de symétrie, on a créé par la méthode précédente des arêtes $A_{i,0} = \langle V, N_i \rangle$ et on a plongé V à l'isobarycentre de N_i. Deux cas de figures sont alors envisageables :

- Il y a un seul N_j qui est aussi sur l'axe de symétrie, ce qui signifie que V est à une extrémité de l'axe de symétrie. On contracte alors l'arête $A_{j,0}$ en N_j.
- N_j et N_k sont tous les deux sur l'axe de symétrie. On contracte alors $A_{j,0}$ en V puis $A_{k,0}$ en N_k. Les trois points se retrouvent maintenant en N_k, je choisis N_k tel que la valeur de Morse associée au contour dont il est le centre soit plus petite que celle du contour dont N_j est le centre.

On pourrait fusionner les points en V mais ce dernier ne correspond pas au centre d'un contour, il est donc plus logique d'utiliser N_k ou N_j qui eux ont un plongement réellement significatif.

La figure 4.10 illustre ces deux cas de simplification et la figure 4.11 donne un exemple de squelette plongé avec cette méthode. Les résultats ne possèdent pas de nœuds non significatifs et ressemblent plus aux squelettes créés et utilisés par les animateurs. Dans les résultats présentés seule la symétrie principale a été utilisée, mais cette simplification peut être menée pour des symétries secondaires.

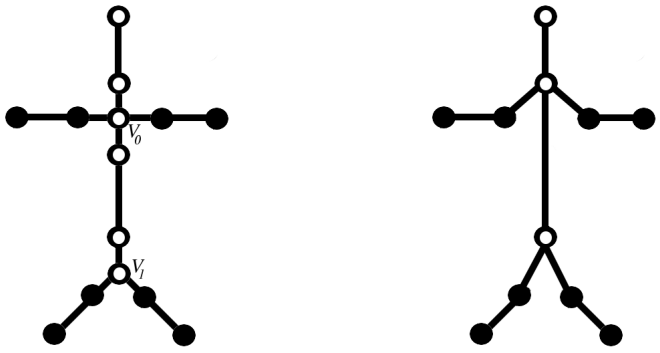
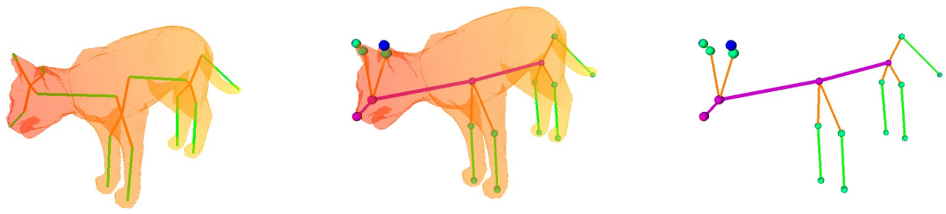


FIG. 4.10 – **(a)** : Le graphe après le plongement simple, les nœuds de l'axe ont un point blanc. V_1 est à une extrémité de l'axe. **(b)** : Graphe simplifié en tenant compte de la symétrie.



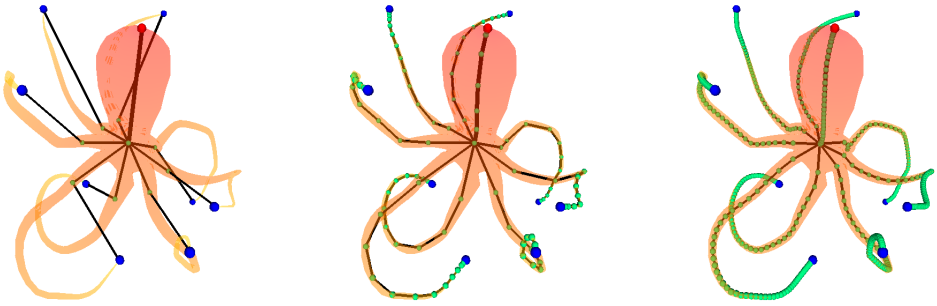
(a) **(b)** **(c)**

FIG. 4.11 – **(a)** : Le squelette après le plongement simple. **(b)** et **(c)** : Le squelette après prise en compte de l'axe de symétrie (représenté en violet).

4.3 Subdivision des arêtes

La figure 4.12 montre sur le modèle de la pieuvre l'importance de subdiviser certaines arêtes pour que le plongement corresponde à la forme. Cette tâche peut être laissée à l'utilisateur qui peut :

- soit fixer un seuil γ tel que le poids de chaque arête du graphe ne soit pas supérieure à γ.
- soit fixer le nombre de nœuds maximum à avoir dans le graphe et subdiviser les arêtes jusqu'à obtenir ce nombre.
- soit sélectionner manuellement les arêtes qu'il souhaite couper en deux, ce qui lui laisse un très grand contrôle sur le squelette obtenu.



(a) **(b)** **(c)**

FIG. 4.12 – **(a)** : Le squelette après plongement en tenant compte de l'axe de rotation mais sans subdiviser. **(b)** et **(c)** : Le squelette après une subdivision plus ou moins fine des arêtes.

Je n'ai implémenté que la manipulation par l'utilisateur mais on peut aussi imaginer des méthodes automatiques :

- diviser chaque arête intersecte la surface du modèle jusqu'à ce qu'elle ne le soit plus. Cette méthode s'adapterait automatiquement à la courbure des branches.
- calculer un grand nombre de contours et leurs centres puis ne conserver que les centres qui présentent des minima locaux en terme d'angle entre les arêtes du graphe plongé. Voir la figure 4.13.

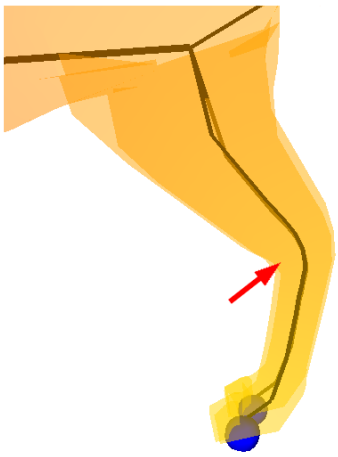


FIG. 4.13 – Détection possible d'une articulation du modèle à un minimum de rayon de courbure du squelette.

4.4 Commentaires des résultats

Nous obtenons des squelettes qui reflètent très fortement la morphologie des modèles. Si le procédé peut être totalement automatisé, l'utilisateur peut cependant choisir de garder un contrôle important sur le résultat : position des extrema, niveau de détail, nombre d'articulations et avec une interface adaptée il peut sélectionner les nœuds ou les arêtes du squelette puis les supprimer, les subdiviser ou encore les déplacer pour ajuster le plongement.

Une poignée de minutes suffisent pour obtenir le squelette souhaité, ce qui représente un gain de temps considérable par rapport aux méthodes de construction de squelettes classiques intégralement manuelles. Les figures 4.14 et 4.15 montrent quelques résultats.

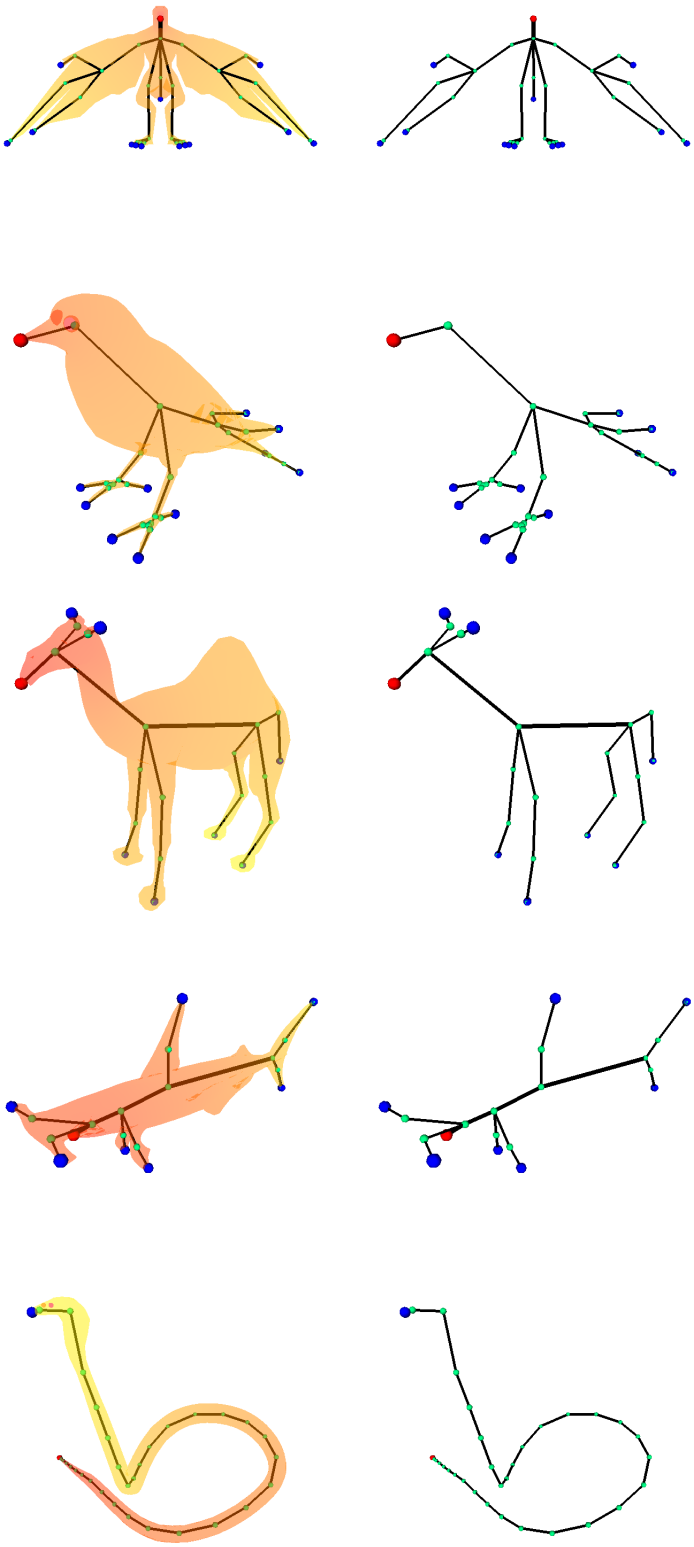


FIG. 4.14 – Modèles et leurs squelettes.

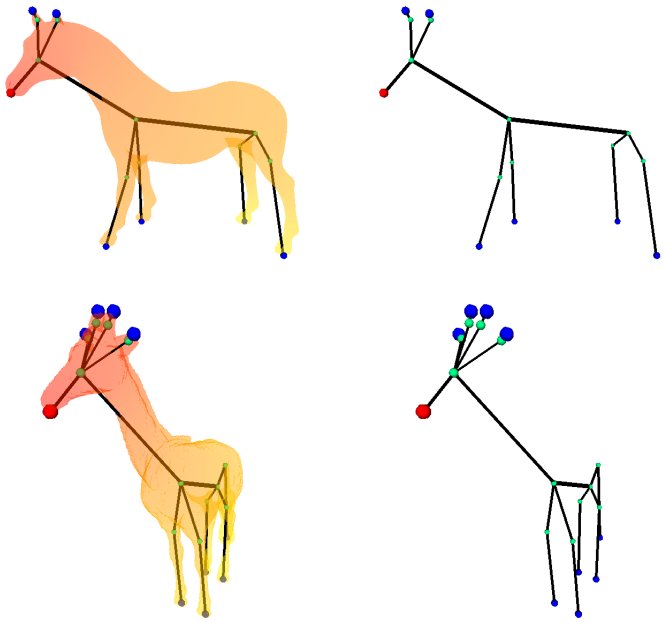


FIG. 4.15 – D'autres modèles et leurs squelettes.

Modèle	**nb. sommets**	**tps de calcul**
ptérodactyle	401	$0s019ms$
oiseau	1129	$0s029ms$
dromadaire	2667	$0s092ms$
requin marteau	2560	$0s108ms$
cobra	4144	$0s157ms$
cheval	48485	$4s227ms$
girafe	68844	$6s127ms$

TAB. 4.1 – Temps de calcul sur différents modèles.

Chapitre 5

Application aux quadrupèdes et aux bipèdes

Il existe dans le domaine de l'animation un squelette type pour les quadrupèdes et un pour les bipèdes. J'ai essayé à partir des squelettes obtenus en plongeant le graphe des contours de fonctions harmoniques de retrouver ces squelettes. Dans la suite de ce chapitre on s'interessera plus particulièrement aux quadrupèdes. Les résultats seront comparés aux squelettes créés par des animateurs professionnels, d'une part visuellement et d'autre part en utilisant les paramètres introduits dans [RFDC05] qui permettent de décrire entièrement le squelette d'un quadrupède.

5.1 Caractérisation des bipèdes et des quadrupèdes

Je décris dans cette partie une méthode pour déterminer si un modèle est un quadrupède ou un bipède. Ils possèdent tous deux un plan de symétrie qui se traduit au niveau du squelette plongé par la même symétrie. Si on impose à l'utilisateur de choisir le point source chaud sur la tête de l'animal, on est ainsi capable de détecter l'axe de symétrie du graphe. On va alors faire quelques suppositions sur le squelette plongé :

- Le squelette est bien celui d'un bipède ou d'un quadrupède, *i.e.* il possède un bassin (nœud B) duquel partent deux pattes/jambes (nœuds P_1 et P_2), des épaules desquelles partent deux pattes/jambes (nœuds A_1 et A_2) et une tête (nœud T).
- Si l'on parcourt l'axe de symétrie en partant du nœud correspondant à la tête du bipède, le dernier point selle rencontré correspond au bassin de l'animal et l'avant dernier à ses épaules.

La figure 5.1 illustre la partie qui nous intéresse de ces squelettes.

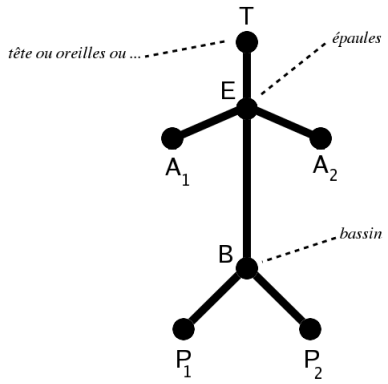


FIG. 5.1 – Squelette d'un bipède ou d'un quadripède.

On considère le graphe plongé, chaque nœud du graphe possède donc des coordonnées dans le repère monde de la scène où se trouve le modèle. Ainsi le vecteur $\vec{AB}$ représente le vecteur entre le plongement des nœuds A et B du graphe. On note alors :

- $\overrightarrow{dos} = \frac{\vec{EB}}{\|\vec{EB}\|}$ le vecteur normé correspondant à la direction de la colonne vertebrale, orientée de l'avant vers l'arrière de l'animal,
- $\vec{N}_P$ un vecteur unitaire normal au triangle BP_1P_2,
- $\vec{N}_A$ un vecteur unitaire normal au triangle EA_1A_2.

D'après les caractéristiques morphologiques de chaque espèce :

- pour un quadrupède on aura : $|\vec{N}_P.\overrightarrow{dos}| \approx 1$ et $|\vec{N}_A.\overrightarrow{dos}| \approx 1$.
- et pour un bipède : $|\vec{N}_P.\overrightarrow{dos}| \approx 0$ et $|\vec{N}_A.\overrightarrow{dos}| \approx 0$.

Cela est vrai pour toutes les positions du modèles, car les nœuds A_1, A_2, P_1 et P_2 correspondent aux débuts des membres qui sont des parties rarement animées (rigidité du tronc).

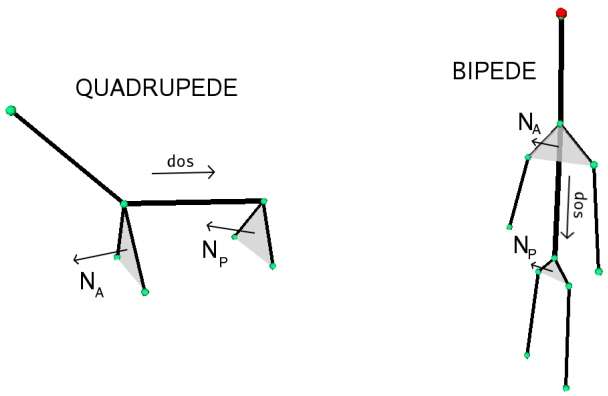


FIG. 5.2 – $\vec{N}_P$ et $\vec{N}_A$ sont parallèles au dos pour un quadrupède et ils lui sont perpendiculaires pour un bipède.

On peut maintenant détecter si le modèle est un quadrupède ou un bipède.

5.2 Plongement spécifique à un quadrupède

En comparant un squelette fait par un animateur professionnel et un squelette généré automatiquement par notre méthode, on note plusieurs points à corriger sur le squelette calculé (voir figure 5.3) :

- la position du bassin est trop basse, il devrait être plus près du dos.
- la colonne vertébrale possède 3 nœuds intermédiaires, situés plus en arrière car cette zone est plus articulable.
- il manque 2 nœuds intermédiaires entre les épaules et le début de chaque patte avant (omoplates).
- il manque 1 nœud intermédiaire entre le bassin et le début de chaque patte arrière (hanches).

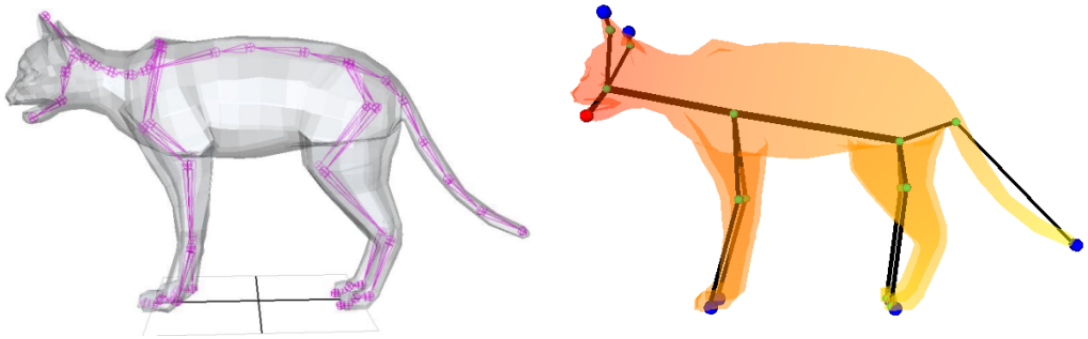


FIG. 5.3 – A gauche un squelette fait par une professionnelle et à droite un squelette calculé.

On s'occupe d'abord de corriger le plongement pour la partie correspondant au bassin de l'animal. Soient :

- $\vec{H_B} = \frac{\vec{P_1B}+\vec{P_2B}}{\|\vec{P_1B}+\vec{P_2B}\|}$ un vecteur unitaire qui donne la direction des pattes arrières vers le haut du dos de l'animal.
- D_B la projection de B sur la surface de l'animal selon la direction $\vec{H_B}$.
- M_i la projection du milieu de $[D_BP_i]$ sur la surface de l'animal selon la direction $\overrightarrow{dos}$.

On en déduit alors un nouveau squelette :

- on déplace le nœud B en D_B ;
- on insère un nœud plongé en M_i sur chaque arête BP_i.

La figure 5.4 illustre ces notations et les changements subis par le squelette sur un schéma :

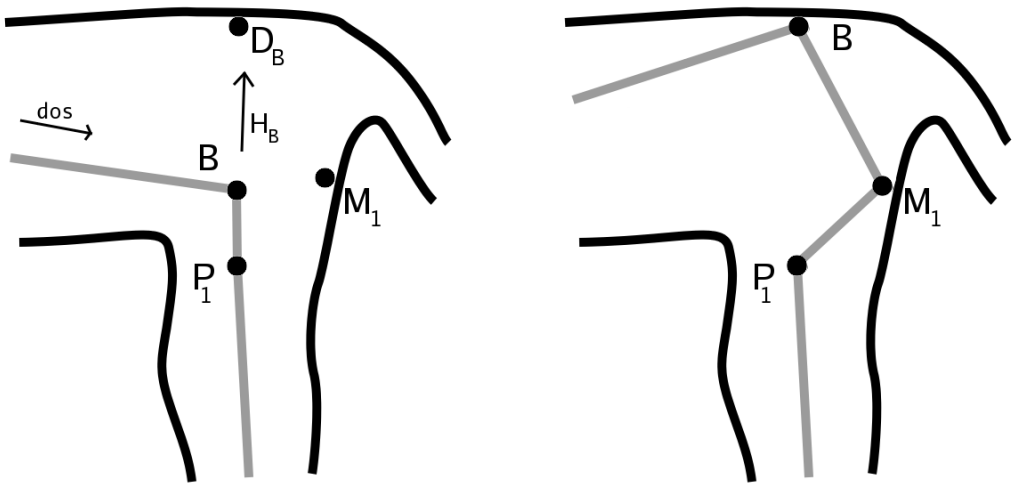


FIG. 5.4 – A gauche le squelette avant modifications et à droite le squelette modifié.

Puis on s'occupe de corriger le plongement pour la partie correspondant aux épaules de l'animal. Soient :

- $\vec{H}_E = \frac{\vec{A_1E}+\vec{A_2E}}{\|\vec{A_1E}+\vec{A_2E}\|}$ un vecteur unitaire qui donne la direction des pattes arrières vers le haut du dos de l'animal.
- D_i la projection de A_i sur la surface de l'animal selon la direction $\vec{H}_E$.
- N_i la projection du milieu de $[A_iD_i]$ sur la surface de l'animal selon la direction $-\overrightarrow{dos}$.

On en déduit alors un nouveau squelette :

- on insère un nœud plongé en D_i sur chaque arête EA_i.
- puis on insère un nœud plongé en N_i sur chaque arête D_iA_i.

Finalement on insère des nœuds sur la colonne vertébrale. Dans les autres cas, les arêtes BP_i et EA_i avaient par construction un poids presque nulle. Or ici la colonne vertébrale a un grand poids et peut être subdivisée en utilisant les valeurs de fonction de Morse de B et E : $f(B)$ et $f(E)$. On insère donc en se basant sur des critères morphologiques de vrais animaux, trois nœuds, C_1, C_2 et C_3 correspondant aux centres des contours associés à la colonne vertébrale et aux valeurs :

- $f(C_1) = f(E) + \frac{1}{2}(f(B) - f(E))$,
- $f(C_2) = f(C_1) + \frac{1}{3}(f(B) - f(C_1))$,
- $f(C_3) = f(C_1) + \frac{2}{3}(f(B) - f(C_1))$.

Remarque : par construction, $f(B) > f(E)$.

La figure 5.5 montre quelques résultats. Sur les conseils d'une animatrice professionnelle travaillant avec des écorchés d'animaux pour aligner ses squelettes d'animation avec les squelettes anatomiques, j'ai ajusté le positionnement de certains nœuds, *e.g.* B ne doit pas être ni collé ni trop loin du dos des animaux.

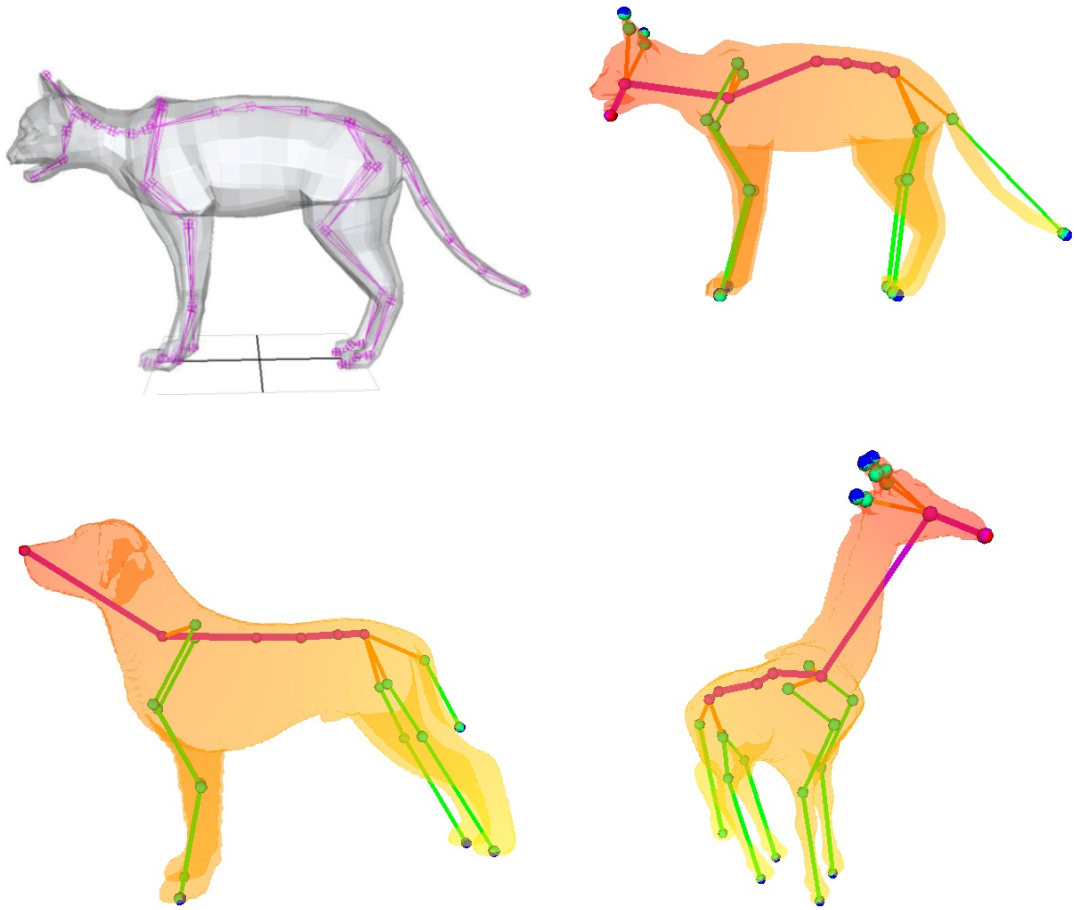


FIG. 5.5 – Squelettes de quadrupèdes obtenus.

5.3 Validation du résultat

La validation des résultats demande plus qu'un simple constat visuel, j'ai donc calculé les paramètres définis dans [RFDC05] afin de caractériser les squelettes d'animation de quadrupèdes, pour les squelettes calculés et je les ai comparés avec ceux des squelettes fait manuellement par un infographiste. La figure 5.6 montre à quoi correspondent ces paramètres. "*spine length*" est la longueur de la colonne vertébrale, "*spine tilt*" correspond à la hauteur au garrot et "*legs height*" est la hauteur des jambes de l'animal (de ses sabots postérieurs jusqu'au haut de sa croupe). On divise la valeur de chacun de ces paramètres par la longueur de la colonne vertébrale qui sert de facteur d'échelle. *spine length* vaut donc toujours 1. Le tableau 5.1 contient les résultats de cette comparaison.

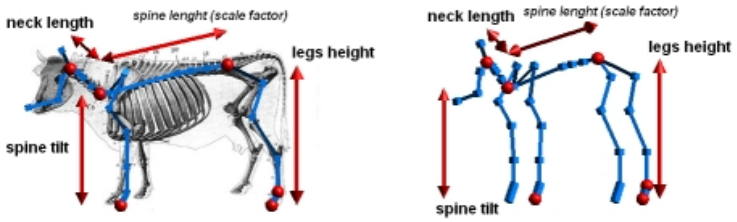


FIG. 5.6 – Paramètres d'un squelette de quadrupède.

Modèle et type de squelette	spine tilt	legs height
chat (calculé)	1.13	1.30
chat (théorique)	1.17	1.23
lionne (calculé)	0.83	1.04
lionne (théorique)	0.89	1.01
chien (calculé)	1.09	1.38
chien (calculé)	1.15	1.45

TAB. 5.1 – Paramètres des squelettes.

Les résultats ne sont pas identiques aux "résultats théoriques" mais n'en sont pas trop éloignés : au maximum 7% de différence. Il m'aurait fallu plus de squelettes théoriques pour mener une étude plus poussée. Les squelettes calculés semblent capturer les bonnes informations sur la morphologie des quadrupèdes et peuvent être utilisés en animation. Ils sont assez proches des squelettes théoriques mais doivent être ajustés manuellement pour un résultat exploitable, ce qui demande très peu de temps. Des plugins pour Maya ou 3DsMax pourraient être réalisés afin de fournir aux animateurs un outil leur permettant de générer automatiquement des squelettes d'animation réalistes. Les résultats obtenus ne permettent pas encore d'imaginer l'automatisation totale de la méthode mais l'intervention de l'animateur reste minime par rapport à la génération manuelle.

Chapitre 6

Animation du modèle grâce au squelette

6.1 Skinning automatique

On appelle skinning le fait de déformer une surface en fonction des mouvements d'un hiérarchie sous-jacente, ici un squelette. Pour cela il faut savoir quelle partie du squelette va déformer quelle partie du modèle. Si le squelette est généré en utilisant le graphe des contours alors on peut avoir une solution immédiate à ce problème. En effet lors de la construction de celui-ci, on balaie tous les sommets du maillage, et pour chaque sommet on sait dire à quelle branche du squelette il appartient. Il suffit donc de stocker pour chaque arête du squelette la liste des sommets du maillage qui lui sont rattachés. Lorsqu'une arête du squelette est déplacée alors les points de la liste le sont aussi en conséquence. La figure 6.1 montre de quelle façon les sommets sont regroupés par branche du squelette.

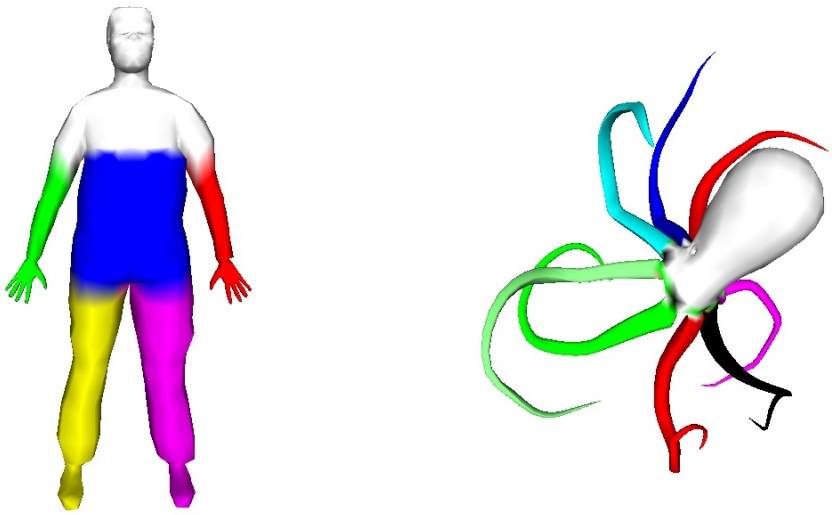


FIG. 6.1 – A chaque branche du squelette est associée une couleur.

Lors des simplifications et du plongement, on a dû supprimer, diviser ou contracter des arêtes, cela se répercute sur le skinning en effectuant des concaténations ou des divisions des listes de sommets entre les arêtes concernées. Pour cela il faut stocker les sommets dans des

listes triées par la valeur de la fonction de Morse des sommets. Ainsi les divisions et les concaténations se font rapidement.

6.2 Essais de déformation axiale

Lazarus [Laz97] a mis au point une méthode permettant d'animer des courbes composées de segments, comme c'est le cas pour un squelette. Cette méthode permet de passer d'une courbe à une autre en minimisant les déformations, et en interpolant le mouvement de manière continue. J'ai utilisé son code afin de tester le skinning généré par la construction du squelette. La figure 6.2 montre un ressort déformé par son squelette généré à l'aide une fonction harmonique.

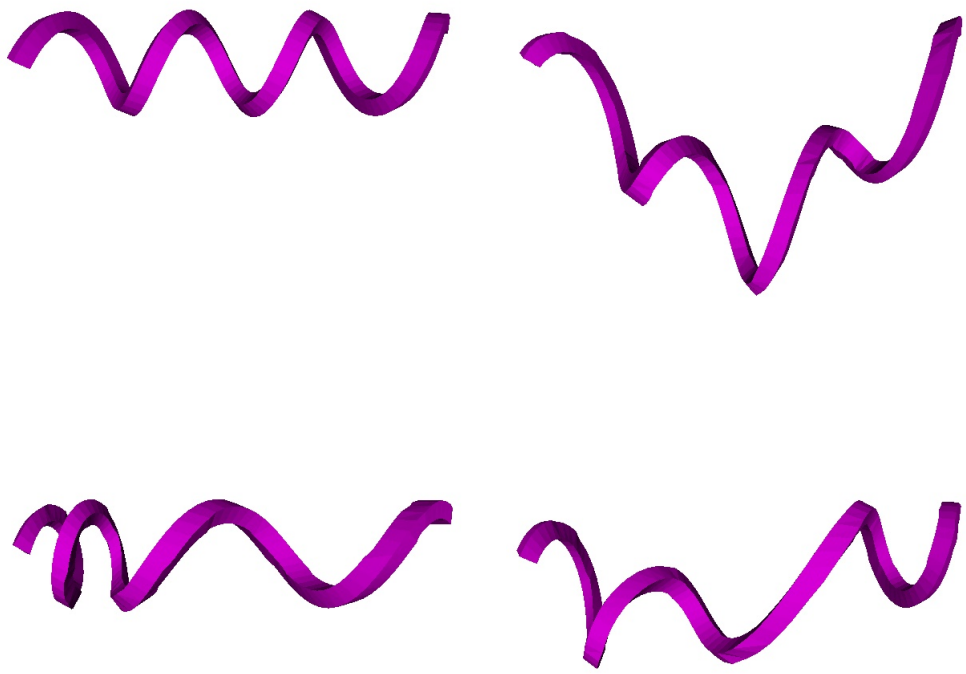


FIG. 6.2 – Déformation d'un ressort utilisant la technique d'interpolation de courbe de Lazarus.

Le résultat est satisfaisant, la tige suit bien son squelette mais il y a des artéfacts au niveau des jonction entre deux arêtes consécutives. Si un triangle possède des points dépendant de deux arêtes différentes alors ce dernier peut être un peu distordu ce qui crée ces artéfacts. En couplant cette technique de déformation axiale avec la génération automatique de squelette, on obtient un outil permettant de déformer n'importe quelle surface triangulée grâce à son squelette.

Chapitre 7

Conclusion et travaux futurs

Dans ce rapport j'ai présenté une méthode permettant de générer des graphes capturant de nombreuses caractéristiques morphologiques des modèles. J'ai de plus présenté une méthode permettant de plonger ces graphes afin de donner des squelettes visuellement proches de squelettes d'animation réalisés par des animateurs professionnels. La méthode de plongement spécifique aux quadrupèdes que je propose a été validée en comparant les paramètres des squelettes à ceux réalisés par une infographiste, ce qui n'avait jamais été fait auparavant.

La principale nouveauté réside dans les simplifications faites aux graphes obtenus par les méthodes existantes. Ces simplifications permettent d'éliminer des caractéristiques inutiles mais aussi de rendre compte de la symétrie du modèle ce qui n'était pas fait jusqu'à présent. Si les points caractéristiques du modèles permettant de fixer les extremités du graphes sont fixés alors celui-ci obtenu est stable et invariant par transformation affine du modèle. Topologiquement, le graphe restera aussi inchangé entre deux positions du modèle lors d'une animation raisonnable du modèle : lever le bras d'un homme ne changera pas le graphe. Il est donc un outil intéressant si l'on cherche à classer des modèles 3D selon leur caractéristiques morphologiques surtout avec l'avènement de bibliothèques d'objets 3D sur Internet. Le procédé de simplification proposé permet en plus de créer une hiérarchie du graphe qui peut être utilisée pour comparer plus facilement deux modèles.

Le plongement des squelettes proposés permet de se rapprocher visuellement des squelettes faits par des animateurs dans des logiciels comme Maya ou 3DS Max pour lesquels la création d'un plugin est envisageable. Un tel outil permettrait un gain de temps important dans le monde de l'animation. Le plongement spécifique aux quadrupèdes que je propose pourrait en être une extension majeure puisque ce sont des modèles très utilisés.

Il reste du travail à accomplir. La détection automatique des points caractéristiques de l'objet n'a pas été suffisamment testée pour dire qu'elle est fiable. Il s'agit d'un point important si l'on veut tendre vers l'automatisation de la méthode de construction du squelette. Cependant je rappelle que le choix manuel de ces points peut être intéressant dans un but de contrôle du résultat.

Les algorithmes de simplifications proposés peuvent être adaptés à des graphes comportant des cycles mais cela demande réflexion.

Il faudrait pousser un peu plus la comparaison avec les squelettes de quadrupèdes créés par des professionnels afin de valider complètement la méthode. Il faudrait aussi confronter des animateurs à l'outil de génération de squelettes pour recueillir leurs critiques afin de l'améliorer.

Enfin, pour l'instant l'animation par déformation du squelette présentée chapitre 6, n'a pas été implémentée pour des objets dont le graphe comporte des points selles : ils n'ont qu'une seule branche comme le ressort. Il faut pour cela réfléchir à ce qui doit se produire aux points selles, comment l'animation d'une branche se répercute sur les autres, etc.

Bibliographie

[AST95] D. Attali, G. Sanniti di Baja, and E. Thiel. Pruning discrete and semicontinuous skeletons. *Lecture Notes in Computer Science*, 974 :488, 1995.

[Ban67] Thomas Banchoff. Critical points and curvature for embedded polyhedra. *Journal of Differential Geometry*, 1 :245–256, 1967.

[Blu67] H. Blum. A transformation for extracting new descriptions of shape. In *Models for the Perception of Speech and Visual Form*, pages 362–380, 1967.

[BMMP03] Silvia Biasotti, Simone Marini, Michela Mortara, and Giuseppe Patanè. An overview on properties and efficacy of topological skeletons in shape modelling. In *Shape Modeling International*, pages 245–256, 297. IEEE Computer Society, 2003.

[BRS03] Dmitriy Bespalov, William C. Regli, and Ali Shokoufandeh. Reeb graph based shape retrieval for CAD, may 2003.

[CMEH+03] Cole-McLaughlin, Edelsbrunner, Harer, Natarajan, and Pascucci. Loops in reeb graphs of 2-manifolds. In *Annual ACM Symposium on Computational Geometry*, 2003.

[ELZ02] Edelsbrunner, Letscher, and Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28, 2002.

[HSKK01] Masaki Hilaga, Yoshihisa Shinagawa, Taku Komura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3D shapes. In *SIGGRAPH*, pages 203–212, 2001.

[KScW+03] L. Kobbelt, P. Schrder, Fu che Wu, Wan chun Ma, Ping chou Liou, Rung huei Liang, and Ming Ouhyoung. Skeleton extraction of 3D objects with visible repulsive force, April 06 2003.

[KT] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. pages 954–961.

[Laz97] Francis Lazarus. Smooth interpolation between two polylines in space. *Computer-aided Design*, 29(3) :189–196, 1997.

[LV99] Francis Lazarus and Anne Verroust. Level set diagrams of polyhedral objects. In *ACM Solid Modeling'99*, Ann Arbor, Michigan, USA, June 1999.

[Mil63] J. Milnor. *Morse Theory*, volume 51 of *Annals of mathematics studies*. Princeton University Press, Princeton, 1963.

[MOR25] M. MORSE. Relations between the critical points of a real functions of n independent variables. *Trans. AMS*, 27 :345–396, 1925.

[MWO03] Wan-Chun Ma, Fu-Che Wu, and Ming Ouhyoung. Skeleton extraction of 3D objects with radial basis functions. In *Shape Modeling International*, pages 207–215, 295. IEEE Computer Society, 2003.

[NGH04] Xinlai Ni, Michael Garland, and John C. Hart. Fair morse functions for extracting the topological structure of a surface mesh. *SIGGRAPH ACM Transactions on Graphics*, 23(3) :613–622, August 2004.

[PSS06] Evangelos Kalogerakis Patricio Simari and Karan Singh. Folding meshes : Hierarchical mesh segmentation based on planar symmetry. In Alla Sheffer Konrad Polthier, editor, *Eurographics Symposium on Geometry Processing*. Eurographics Association, 2006.

[Ree46] G. Reeb. Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique. *Comptes-rendus de l'Académie des Sciences*, 222 :847–849, 1946.

[RFDC05] Lionel Reveret, Laurent Favreau, Christine Depraz, and Marie-Paule Cani. Morphable model of quadrupeds skeletons for animating 3D animals. In Demetri Terzopoulos and Victor Zordan, editors, *ACM SIGGRAPH /Eurographics Symposium on Computer Animation*, pages 135–142, Los Angeles, California, 2005. Eurographics Association.

[SDG05] S. Kircher S. Dong and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometry Design, Special Issue on Geometry Processing*, 22 :392–423, 2005.

[TDV06] Julien Tierny, Mohamed Daoudi, and Jean-Philippe Vandeborre. Invariant high-level reeb graphs of 3d polygonal meshes. Chapel Hill, North Carolina, USA, June 2006. 3rd IEEE International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'06).

[TS04] Tony Tung and Francis Schmitt. Augmented reeb graphs for content-based retrieval of 3D mesh models. In *Shape Modeling International*, pages 157–166. IEEE Computer Society, 2004.

[vKvOB+97] van Kreveld, van Oostrum, Bajaj, Pascucci, and Schikore. Contour trees and small seed sets for isosurface traversal. In *Annual ACM Symposium on Computational Geometry*, 1997.

[WP02] Lawson Wade and Richard E. Parent. Automated generation of control skeletons for use in animation. *The Visual Computer*, 18(2) :97–110, 2002.