Digital geometry and algorithmic geometry for interactive 3D design

Jean-Marc Thiery

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2012-ENST-070

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « SIGNAL et IMAGES »

présentée et soutenue publiquement par

Jean-Marc THIERY

le 28 novembre 2012

Géométrie numérique et géométrie algorithmique pour le design intéractif 3D

Directeur de thèse : **Tamy BOUBEKEUR**

Jury

M. Bruno LEVY, Directeur de recherche, LIRIA, INRIA Nancy Grand-Est	Rapporteur
M. Marc ALEXA, Professeur, TU Berlin	Rapporteur
M. Henri MAÎTRE, Professeur, LTCI, Télécom-ParisTech	Examinateur
M. Gabriel PEYRE, Chargé de recherche, CNRS, Univ. Paris Dauphine	Examinateur
M. Tamy BOUBEKEUR, Maître de conférence (HDR), CNRS-LTCI, Télécom-ParisTech	Directeur

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

THÈSE

Géométrie numérique et géométrie algorithmique pour le design intéractif 3D

Digital geometry and algorithmic geometry for interactive 3D design

Jean-Marc Thiery

Remerciements

Il m'est probablement impossible d'énumérer les personnes que je devrais remercier pour toutes les choses qu'elles ont faites, et qui ont créé la suite d'évènements improbables qui m'ont finalement amené à être en train de rédiger les remerciements de ma thèse. Au moment de le faire, je ne sais plus. Je vais donc me contenter de remercier les amis que j'ai pu découvrir pendant cette thèse, et particulièrement les camarades qui ont partagé mes soucis quotidiens: Noura, Matthias, Elmar, Tamy, Bert, Jing, Guillaume, Julien, et d'autres... Je veux également remercier ma bien aimée Claudia qui m'a supporté au mieux pendant cette période riche en stress et frustration.

D'un autre côté, il y a un certain nombre de choses dans ma vie pour lesquelles je devrais faire preuve de gratitude, mais la plupart n'ayant aucun lien avec cette thèse, le bon goût m'interdit de m'épancher sur celles-ci à cette occasion.

Il y a enfin quelques personnes qui ont eu une grande influence sur ma vie, que j'ai vraiment envie de remercier, en laissant une preuve écrite qu'elles pourront ressortir d'une étagère de temps à autre. Je veux remercier ces personnes pour les efforts qu'elles ont fournis, et je sais qu'elles ont fait au mieux en toute occasion. Ces personnes sont ma famille, mon défun père, ma mère, mon grand père, mon grand frère, ma grande soeur, mes petits frères.

While 3D surfaces are essentially represented using triangle meshes in the domain of digital geometry, the structures that allow to interact with those are various and adapted to the different geometry processing tasks that are targetted by the user.

This thesis presents results on structures of various dimension and various geometrical representations, going from internal structures like analytical curve skeletons for shape modeling, to on-surface structures allowing automatic selection of feature handles for shape deformation, and external control structures known as “cages” offering a high-level representation of animated 3D data stemming from performance capture. Results on spatial functions are also presented, in particular for the Mean-Value Coordinates, for which the analytical formulae of the gradients and the Hessians are provided, and biharmonic functions, for which a finite elements basis is given for the resolution of the biharmonic Laplace problem with mixed Dirichlet/Neumann boundary conditions, as well as their applications to 3D shapes deformation.

Alors que les surfaces géométriques sont essentiellement représentées à l’aide de maillages triangulaires dans le domaine de la géométrie numérique, les structures permettant d’interagir avec ces géométries sont variées et adaptées aux différents traitements visés par l’utilisateur.

Cette thèse présente des travaux réalisés sur des structures de dimension et de représentation géométrique variées, allant de l’étude des structures internes comme les squelettes analytiques pour la modélisation géométrique, passant par les structures surfaciques pour la sélection automatiques de poignées de déformation, jusqu’aux structures externes de contrôle d’objet de type "cage" offrant une représentation haut niveau de séquences animées d’objets issues de systèmes de performance capture. Sont présentés également les résultats obtenus sur les coordonnées aux valeurs moyennes offrant une solution au problème de l’interpolation de conditions de Dirichlet, pour lesquelles les formules analytiques des gradients et Hessiens sont fournies, et les fonctions biharmoniques pour lesquelles une base d’éléments finis est formulée pour la résolution du problème de Laplace biharmonique avec conditions mixtes Dirichlet/Neumann, ainsi que leurs applications à la déformation de formes 3D.

Contents

Chapter 1

Long résumé en langue française

L'objectif de cette thèse est de développer de nouvelles méthodes de design 3D interactif s'appuyant sur les outils de géométrie numérique et géométrie algorithmique afin d'offrir un contrôle de plus haut niveau dans les étapes de création et d'édition d'objets 3D.

Dans ce cadre, j'ai essentiellement étudié les structures topologiques et géométriques permettant l'itéraction avec des formes 3D complexes, et j'ai développé des algorithmes qui ont permis la création ou l'utilisation de celles-ci. J'ai classifié ces structures en trois parties: 1) les structures internes (squelettes et axes médians), 2) les structures surfaciques et locales (courbes sur la surface de l'objet, "patches"), et 3) les structures externes (cages et boites englobantes).

1.1 Contributions techniques

Concernant les structures internes, j'ai développé un algorithme de création de squelette unidimensionnel *analytique*, permettant de réaliser des traitements géométriques intuitifs sur les objets présentant des structures tubulaires. Cet algorithme est typiquement adapté à la décomposition de personnages (humains, animaux, ...). Les traitements introduits sont: 1) la création assistée de "surfaces internes" (création de volumes à l'intérieur de la surface) et l'application au "shell mapping", 2) l'édition de formes basée sur la décomposition de l'objet sur son squelette, et 3) le filtrage bilatéral basé encore une fois sur cette décomposition.

Concernant les structures surfaciques, j'ai étudié avec ma collègue Leila Schemali comment définir des courbes (à partir de la géométrie de l'objet ainsi que du point de vue) sur l'objet 3D, pour permettre à l'utilisateur de les manipuler et ainsi de déformer sa surface. Avant cela, je m'étais intéressé à la définition *multi-résolution* d'une structure de *complexe* sur l'objet, permettant à l'utilisateur de sélectionner des *régions* de l'objet, ou bien des courbes séparant ces régions, ou encore des points intersectants ces différentes courbes. Une interface utilisateur a été définie pour que l'utilisateur puisse, en temps-réel, passer de l'édition surfacique de grandes régions à l'édition locale de détails géométriques.

Concernant les structures externes, je me suis intéressé à l' *ingénieurie inverse* d'animations d'objets 3D à l'aide de cages. Dans ce contexte, l'artiste dispose de l'animation d'un objet 3D complexe, et il souhaite trouver la déformation d'une cage générant l'animation de cet objet. Cela permet de *paramétrer* simplement l'animation, et cela permet les applications suivantes: 1) la compression d'animations à l'aide d'une structure *naturelle* de manipulation de volumes, 2) l'édition intéractive de l'animation, 3) l'accélération d'algorithmes géométriques complexes (comme l'interpolation générales de formes 3D) en utilisant les cages produites comme des versions basse résolution de l'objet déformé.

Je me suis également intéressé à l'étude de coordonnées de cages dans leur définition mathématique. Dans ce cadre, j'ai obtenu les formules analytiques des dérivées au premier ordre (Jacobien) et deuxième ordre (Hessien) des coordonnées aux valeurs moyennes (*Mean Value Coordinates*) introduites par Floater. J'ai également travaillé à l'implémentation d'un algorithme permettant l'obtention par des méthodes numérique d'approximation des dérivées des coordonnées *harmoniques*. Dans les deux cas, j'ai étudié les applications possibles, et notamment la définition de systèmes de modélisation *implicites* à l'aide de cages, permettant l'optimisation de la géométrie de la cage en fonction de contraintes positionnelles placées directement sur l'objet déformé, ainsi que de contraintes de rotation (Jacobiens contraints à être des matrices de rotation dont on ne connait pas la valeur à priori) et des contraintes de rigidité (Hessiens contraints à avoir une norme minimale). Avec ce formalisme, l'utilisateur itéragit *directement* avec l'objet qu'il déforme, la cage ne servant que de support pour la déformation *en arrière plan*, et il peut à tout moment modifier directement la cage pour rajouter des détails de manière locale.
En plus de l'étude des coordonnées aux valeurs moyennes, j'ai également obtenu les formules analytiques d'une base d'éléments finis pour les fonctions biharmoniques.

J'ai également participé à divers travaux en collaboration avec mes collègues thésards, dans le domaîne de l'imagerie médicale avec Noura Faraj, dans le domaîne de la création et paramétrisation de maillages avec Guillaume Vialaneix, et dans le domaîne de la modélisation 3D avec Leila Schemali et Bert Buchholz.

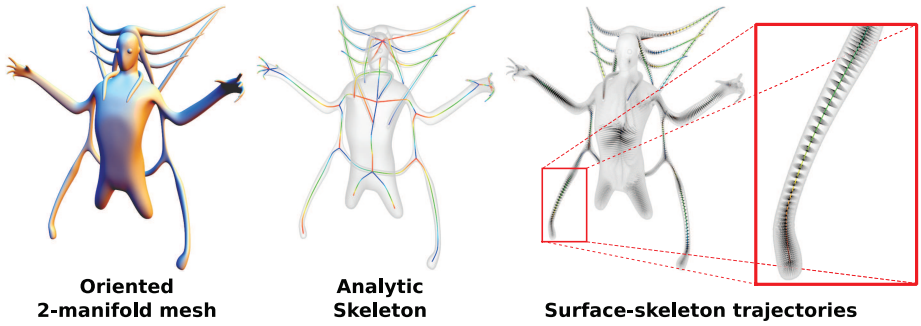


Figure 1.1: **Squelette analytique** d'une surface polygonale. La définition d'une relation surface-squelette avec peu de variation offre plusieurs applications au traitement du signal et à la modélisation *squelettale* de surfaces 3D.

1.2 Création de squelettes analytiques pour la modélisation

Dans cette première partie, on présente un algorithme de création de squelette analytique à partir de surfaces closes en 3D. Décomposer une forme 3D sur un domaîne plus simple pour effectuer des traitements géométriques n'est pas une idée nouvelle, et quelques personnes ont du essayer de le faire en prenant comme domaîne de décomposition un squelette (bien que la bibliographie à ce sujet soit inexistante).

La nouveauté introduite dans ce travail est la décomposition naturelle de l'objet et de son squelette, qui sont deux objets optimisés conjointement, ainsi que la forme analytique du squelette, qui permet de considérer la relation surface-squelette partout sur la variété, et de manière régulière. La stratégie adoptée est la suivante: **(i)** décomposer la surface initiale en cylindres topologiques, **(ii)** intégrer géométriquement chaque cylindre pour trouver une courbe en son milieu (qui est un *os* du squelette créé), et **(iii)** optimiser la relation surface-squelette grace à un algorithme de reprojection adapté de la forme sur son squelette.

Le résultat de cet algorithme est illustré en figure 1.1, où on peut observer que les trajectoires surface-squelette sont partout définies et contiennent peu de variations, permettant de considérer ces trajectoires comme un champ de déplacement sur la surface adapté aux applications de *"déplacement"* (ces modifications étant réalisées traditionnellement le long de la normale à la surface, introduisant rapidement des auto-intersections de la surface, même pour des "petits" déplacements).

1.2.1 Décomposition de surfaces en cylindres et disques topologiques

La décomposition de la surface d'entrée s'appuie sur un algorithme de réduction de graphe. Initialement, chaque triangle t_i de la surface est une région R^i, et le graphe d'adjacence des régions est construit. Puis, à chaque étape de la réduction de ce graphe par contraction d'arête, deux régions adjacentes R^A et R^B peuvent être unies, détruisant au passage une région (on note cette opération $R^{A \cup B} = R^A \cup R^B$). L'ordre dans lequel les régions sont aggrégées détermine de manière unique la topologie de la segmentation résultante.

Topologie			Nom	e_T
R^A	R^B	$R^A \cup R^B$		
Cyl.	Disque	Disque	*Fill Hole*	**0**
Disque	Disque	Cyl.	*Create Cylinder*	**1**
Disque	Disque	Disque	*Merge Disks*	**2**
Cyl.	Disque	Cyl.	*Grow Cylinder*	**3**
Cyl.	Cyl.	Cyl.	*Merge Cylinders*	**4**
Autres cas			*Skip*	∞

Table 1.1: Terme de priorité topologique.

Traditionnellement, ce genre d'algorithme a été utilisé pour la simplification de maillages [37], où uniquement la géométrie des régions détermine l'ordonnancement des réductions successives. La nouveauté technique que nous introduisons ici est le terme d'erreur associé à l'étape de réduction $R^{A\cup B} = R^A \cup R^B$, pour lequel la topologie des trois régions est discriminante (voir Table 1.1).

Les aggrégations de régions sont ordonnées premièrement par ordre croissant de *coût topologique* e_T, puis par ordre croissant de coût géométrique e_G au sein d'une même classe topologique. La topologie des différentes régions peut être contrôlée à l'aide de la *caractéristique d'Euler* χ (nombre de sommets moins le nombre d'arêtes, plus le nombre de triangles), qui vaut 1 pour un disque topologique et 0 pour un cylindre topologique. Pour réaliser ces tests de manière efficace, une structure de "méta"-maillages (composés de sommets, courbes, et régions, en lieu et place d'arêtes et de triangles) peut être utilisée, car la caractéristique d'Euler est un invariant topologique, et ne dépend donc pas de la représentation particulière des surfaces considérées.

Terme géométrique Les termes géométriques que nous considérons permettent de favoriser l'expansion des régions convexes. Dans la suite, on note $s^X = \int_{\xi\in X} d\sigma_\xi$ la surface totale de la région $R^X = R^A \cup R^B$, $\mathbf{n}^X = \int_{\xi\in X} n_\xi d\sigma_\xi / s^X$ sa normale moyenne, et $\mathbf{p}^X = \int_{\xi\in X} p_\xi d\sigma_\xi / s^X$ sa position moyenne.

L'erreur géométrique $e_G(A,B)$ associée à l'aggrégation des région R^A et R^B dépend de la classe topologique $e_T(A,B)$, et on la définit de manière formelle comme $e_G(A,B) = E_{e_T(A,B)}(A,B)$.

$$E_0(A,B) = 1 - \|\mathbf{n}^X\| \tag{1.1}$$

$$E_1(A,B) = E_4(A,B) = \|\mathbf{n}^X\| \tag{1.2}$$

$$E_2(A,B) = E_3(A,B) = \sum_i s_i \cdot \|\mathbf{p}_i - \mathbf{p}^X\|^2 / s^X \tag{1.3}$$

Minimiser $E_0(A,B)$ revient à avoir une normale moyenne ayant une norme proche de 1, ce qui n'est possible que si la région est planaire: E_0 sert donc à effacer les cylindres topologiques qui n'ont pas de sens géométrique ("reboucher les trous").

Figure 1.2: Résultats de segmentation de surfaces en disques et cylindres topologiques.

Comme E_1 doit favoriser la création de cylindres et que E_4 doit détecter l'union possible de cylindres créant un cylindre valide, on modélise l'erreur E_1 comme une normale moyenne de norme minimale (au contraire de E_0). On utilise ici le fait que des cylindres géométriques parfaits ont une normale moyenne nulle.

Les erreurs géometriques E_2 (*Merge Disks*) et E_3 (*Grow Cylinder*) sont définies comme les déviations standards géométriques, dont les minimiseurs sont les formes sphériques.

Prédicat de validité Pour éviter de réduire trop la segmentation de la surface et d'effectuer des aggrégations de régions menant à des régions insatisfaisantes, il est essentiel d'interdire l'apparition de certaines configurations.

Le premier critère est d'ordre topologique, et est lié directement à l'objet que l'on cherche à créer: on interdit l'apparition de patches ayant une topologie différente d'un disque ou d'un cylindre (voir Table 1.1). Les autres critères sont essentiellement géométriques, et sont expliqués plus en détail dans le chapitre 4.

Des résultats de décomposition peuvent être observés en figure 1.2.

1.2.2 Intégration géométrique

Une fois que l'objet est décomposé en disques et cylindres topologiques, il est trivial d'en extraire un squelette en construisant pour chaque cylindre un "os", qui prend dans notre cas la forme d'une courbe en 3D. Il est à noter que tous les sommets d'un disque sont projetés sur l'extrémité de l'os correspondant au cylindre voisin dans la segmentation.

Pour cela, on paramétrise chaque cylindre C_i à l'aide d'une carte $u : C_i \rightarrow [0, 1]$ *harmonique*, en résolvant le système de Laplace suivant: $u = 0$ sur un bord du cylindre, $u = 1$ sur l'autre bord, et $\triangle\ u = 0$ sur les sommets à l'intérieur du cylindre. Comme nous travaillons sur des surfaces triangulaires, nous utilisons une discrétisation de l'opérateur Laplacien, en l'occurence celle dites des "poids cotangents" (voir chapitre 3). Le système est un système linéaire carré de taille $|v(C_i)|$ (le nombre de sommets appartenant au cylindre C_i), que l'on peut résoudre au sens des moindres carrés (notre implémentation repose sur la librairie Cholmod).

Une fois que nous avons obtenu la coordonnée u sur le cylindre (u étant une coordonnée longitudinale), on peut compléter cette carte unidimensionnelle en une carte bidimensionnelle (u, θ) (θ étant une coordonnée cylindrique) en coupant le cylindre C_i le long du chemin qui est le plus aligné possible avec le gradient de la coordonnée longitudinale $\overrightarrow{\nabla} u$, et en répétant le procédé pour obtenir une carte harmonique θ qui prend les valeurs 0 et 2π sur les deux bords ainsi obtenus.

Avec cette carte $(u, \theta) : C_i \to [0, 1] \times [0, 2\pi]$, il est possible de contracter chaque isocourbe $u = \hat{u}$ du cylindre en un point $b_i(\hat{u})$ à l'aide de la formule $b_i(\hat{u}) = \frac{\int_0^{2\pi} p(\hat{u},\theta) d\theta}{2\pi}$, construisant ainsi la fonction géométrique $b_i : [0, 1] \to \mathbb{R}^3$ qui décrit de manière paramétrique l'os correspondant au cylindre C_i.

1.2.3 Itération à l'aide de reprojections paramétriques

Il a déjà été remarqué que la restriction à la surface initiale du diagramme de Voronoï des os d'un squelette tendait à créer des cylindres associés à chaque os sur la surface (ce procédé consiste à segmenter la surface d'origine en attribuant à chaque sommet de la surface l'index de l'os le plus proche). Les cylindres étant issus du diagramme de Voronoï, leurs frontières sur l'objet sont régulières, ce qui améliore la qualité de la segmentation et du squelette qui en est issus de manière significative. Le lecteur peut remarquer que jusqu'à présent, la géométrie des *frontières* des cylindres n'a pas été contrainte, et que les résultats présentés en figure 1.2 contiennent des cylindres aux frontières irrégulières. L'optimisation proposée est donc d'utiliser la segmentation de la surface issue du diagramme de Voronoï des os comme segmentation d'entrée de l'algorithme décrit, pour obtenir au fil des itérations des cylindres avec des frontières de plus en plus régulières.

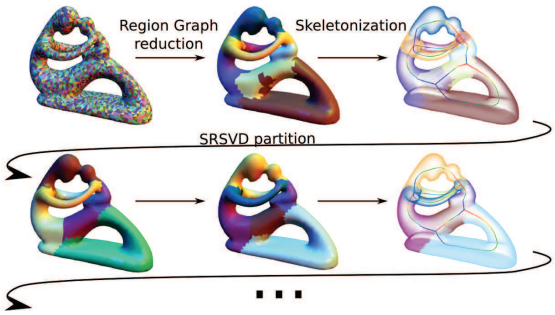
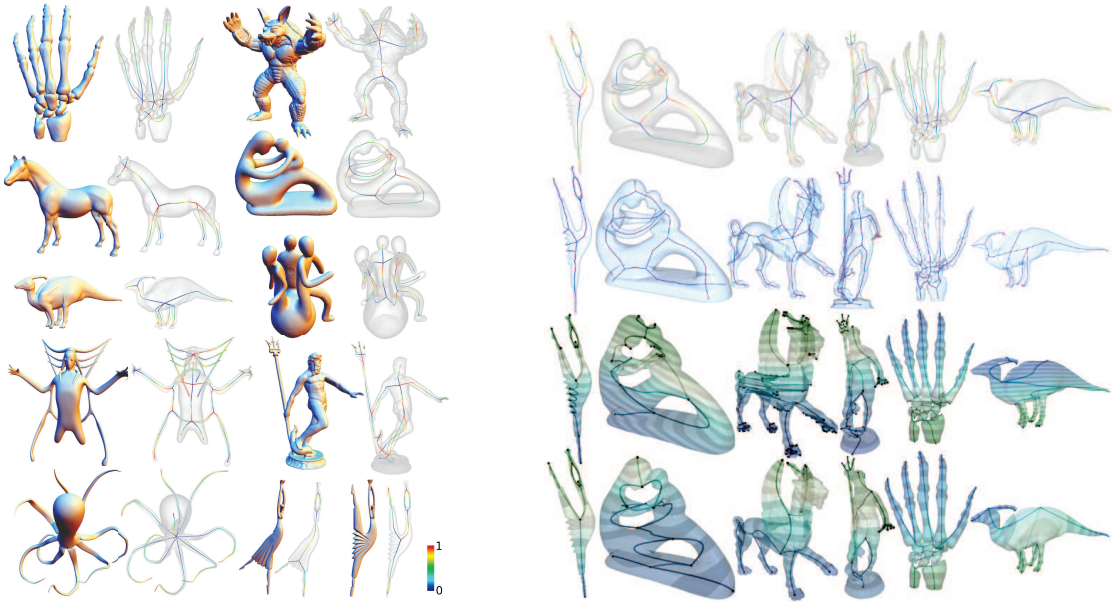


Figure 1.3: Chaque ligne présente une itération de l'algorithme présenté. L'entrée de l'itération suivante est définie à l'aide de la reprojection paramétrique de la forme sur son squelette.

Cependant, cette stratégie permet d'obtenir une segmentation de la surface d'origine composée de cylindres et de disques uniquement dans un nombre extrèmement restreint de cas. La plupart du temps, l'objet 3D a une épaisseur (distance à l'os, ou à l'axe médian) qui varie, et cette variation introduit des cas pathologiques dans la segmentation qui résulte de la reprojection sur le squelette.

Une manière de contourner ce problème est de paramétriser la reprojection de la surface sur le squelette à l'aide d'un unique paramètre α ayant une valeur entre 0 et 1, et qui décrit à quel point on translate la surface d'origine vers le squelette. Plus précisemment, si les sommets v du maillage d'entrée sont projetés sur le squelette au point $b(v) \in \mathbb{R}^3$, on crée son translaté $\hat{v}_\alpha \in \mathbb{R}^3$ de la manière suivante: $\forall v : \hat{v}_\alpha = \alpha \cdot b(v) + (1 - \alpha) \cdot v$. La surface ainsi définie est donc "plus proche" du squelette, et la restriction du diagramme de Voronoï



(a) Résultats (b) Comparaison

Figure 1.4: (a): Squelettes obtenus pour divers objets 3D. Le code couleur indique la paramétrisation de chaque os. (b): Comparaison avec les squelettes Laplaciens [4] (deuxième ligne), et des graphes de Reeb de fonctions hauteurs (troisième ligne) et harmoniques (quatrième ligne).

du squelette à cette surface paramétrique donne des cylindres ayant une topologie valide lorsque α est pris assez grand (dans le pire des cas, pour $\alpha = 1$, la surface d'origine est entièrement projetée sur le squelette, et la segmentation obtenue à l'aide du diagramme de Voronoï est donc identique à celle de l'itération précédente).

1.2.4 Résultats

Des résultats de la technique introduite sont présentés en figure 1.4(a), où il peut être vérifié visuellement que (i) la décomposition des objets est "celle que l'on pouvait attendre", et que (ii) les squelettes trouvés ont une géométrie régulière.

Il est à noter que l'objectif principal de cette technique est de fournir un squelette ayant une géométrie la plus régulière possible, et qu'il n'est pas contraint de manière explicite à être situé "à l'intérieur de l'objet 3D". La raison de ce choix est la classe d'applications visée, qui est le traitement géométrique et la modélisation géométrique à l'aide de squelettes. Pour la plupart de ces applications, il est plus important d'avoir une contrainte "dure" sur la régularité du squelette et une contrainte "douce" sur la localisation de celui-ci près de l'axe médian, plutôt que l'inverse.

Il faut également remarquer que ces deux contraintes sont en opposition: seul l'axe médian est une représentation qui est localisée parfaitement à l'intérieur et au centre de l'objet, et par définition cet objet est géométriquement irrégulier (il correspond à l'ensemble des points pour lesquels le gradient de la distance à l'objet n'est pas défini).

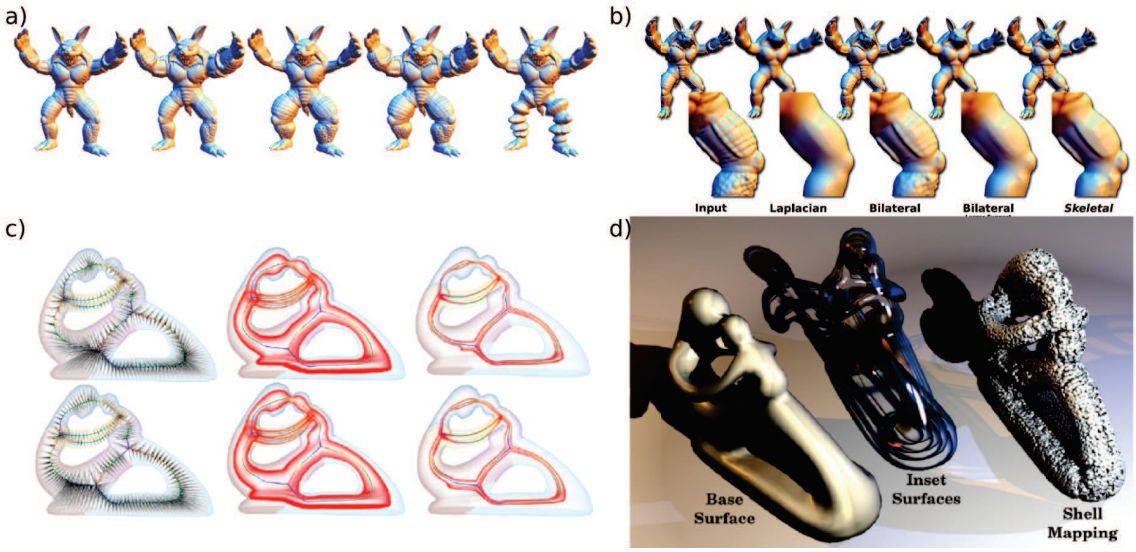


Figure 1.5: **a)** Variations obtenues à l'aide de cartes de déplacement le long des trajectoires. **b)** Filtrage bilatéral, utilisant le squelette comme signal. **c)** Création de surfaces internes paramétriques, le long des trajectoires. **d)** Application au "Shell Mapping": rendu de particules et de fonctions de densité dans le volume créé.

La figure 1.4(b) présente des éléments de comparaison avec les techniques les plus utilisées actuellement pour obtenir des squelettes unidimensionnels, qui sont basées sur l'extraction du graphe de Reeb de fonctions scalaires ad-hoc ou sur la contraction de surfaces à l'aide de son opérateur Laplacien [4].

1.2.5 Applications

La figure 1.5 illustre des applications d'un squelette analytique régulier. Ces applications tirent toutes parti des propriétés mathématique de notre modèle de squelette.

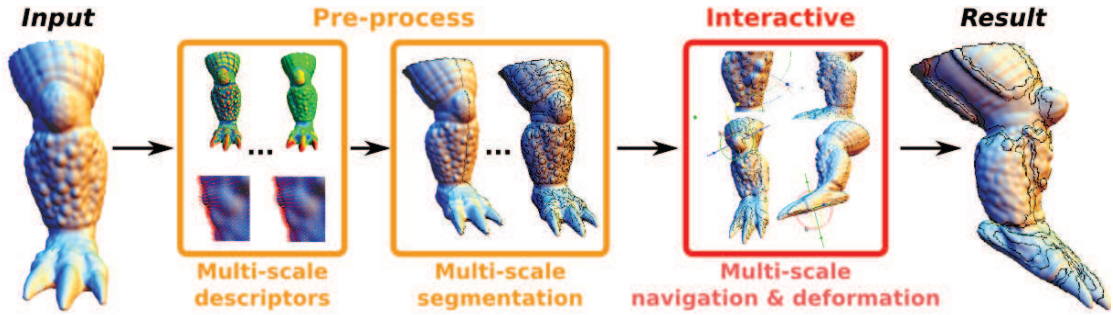


Figure 1.6: Framework de DEX

1.3 Détection automatiques de structures surfaciques pour la manipulation

Dans la première partie de cette thèse, on a présenté des résultats et des applications de modélisation basée sur une structure *interne* unidimensionnelle.

Nous nous intéressons dans cette deuxième partie aux structures surfaciques qu'il est possible d'extraire automatiquement à partir d'un objet 3D, dans le contexte de la manipulation directe d'objet, à l'aide de métaphores de modélisation faisant intervenir la minimisation d'énergie géométriques (au contraire de modélisation physiques).

Cette classe de techniques de déformation d'objets 3D est répandue dans la communauté graphique, et nombre d'entre elles reposent sur une stratégie commune pour la modélisation: L'artiste, lorsqu'il veut déformer l'objet, définit une partie de celui-ci qu'il *attrape* à l'aide d'un manipulateur 3D, ainsi qu'une *zone d'influence* dont la géométrie est optimisée pour minimiser différentes énergies géométriques, tandis que le reste de l'objet reste inchangé. La définition même de ces énergies géométriques a été traité très largement dans la littérature, et nombre de techniques sont suffisamment satisfaisantes pour les artistes pour être utilisées courrament dans l'industrie du film d'animation.

La problématique que nous soulevons ici est que la définition des zones d'influence pour l'application de ces techniques est une tache fastidieuse, qui constitue en fait la majeure partie du temps d'une session de modélisation. Nous proposons donc d'automatiser ce processus en permettant diverses manières de définir des zones intéressantes à manipuler par l'artiste.

1.3.1 Complexe de déformation

La première structure de manipulation que nous proposons est un *complexe* de déformation multi-résolution sur la surface: La surface d'origine est segmentée en *patches* que l'utilisateur peut attraper. Deux patches voisins sont séparées par des *courbes* qui sont composées d'arêtes des triangles de la surface d'origine, et l'intersection non vide de deux courbes sont des *sommets*. L'ensemble de la structure ainsi définie est donc un complexe simplicial calculé avec lequel l'artiste peut intéragir.

La figure 1.6 présente le framework la construction du complexe simplicial de déformation, appelé *DEX*. (Pré-calcul:) un ensemble de descripteurs multi-resolution est premièrement calculé sur la surface d'origine, et une segmentation multi-résolution est effectuée à partir de ceux-ci; (intéraction:) l'utilisateur peut alors naviguer dans les différents niveaux de hiérarchie et sélectionner l'ensemble des parties pour la déformation, la zone d'influence étant calculée automatiquement à partir des relations de voisinage dans le complexe surfacique.

Les descripteurs que nous utilisons ici sont basés sur les informations de normales de la surface (dérivée du premier ordre de la géométrie) et de courbures (dérivée du second ordre de la géométrie). Leur construction est illustrée en figure 1.7: ceux-ci sont calculés au plus haut niveau de la hiérarchie à partir de la représentation triangulaire de la surface, avant d'être *moyennés* avec des noyaux de taille de plus en plus grande dans les niveaux de hiérarchie les plus bas.

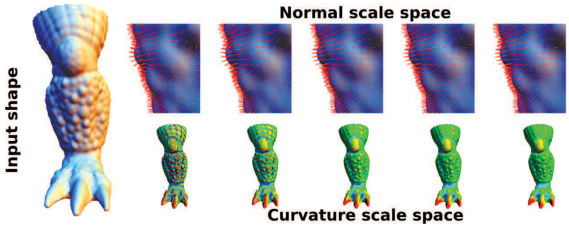


Figure 1.7: Descripteurs multi-résolution.

A partir de ces descripteurs, une segmentation est effectuée à chaque niveau de la hiérarchie, avec de moins en moins de parties tout au long de la hiérarchie, en se basant essentiellement sur la normale aux bas niveaux et sur la courbure aux hauts niveaux. L'algorithme utilisé pour la segmentation est une variante de l'algorithme "k-means", qui minimise au sein de chaque partie de la segmentation la variance du descripteur utilisé. Cette variante garantit par construction que chaque ensemble dans la segmentation est simplement connectée sur la surface.

Une session de modélisation utilisant le complexe de déformation est illustrée figure 1.8. L'utilisateur édite des caractéristiques géométriques du maillage sélectionnées par notre approche multi-résolution, l'ordre dans lequel il choisit les différents niveaux de hiérarchie pour les étapes successives de la session d'édition ne lui étant pas imposé par notre technique.

Discussion Premièrement, puisque nous construisons le complexe de déformation à partir d'une segmentation de l'objet, nous faisons l'hypothèse que chaque partie de l'objet peut être d'un certain intérêt pour la sélection de poignées de déformation. Cette hypothèse n'est pas tout le temps vérifiée, mais notre formulation est utile pour une large classes d'objets 3D contenant des caractéristiques d'ordres de grandeur différents.

Deuxièmement, les énergies que nous minimisons sont limités à la représentation de certaines classes de zones caractéristiques, qui sont des zones quasi-planaires à grosse échelle et les extrusions locales et variations rapides de la surface à une échelle plus fine.

Dans la partie suivante, on étudie une approche complémentaire et montrons comment des courbes caractéristiques peuvent aider à la définition automatique de poignées de déformation.

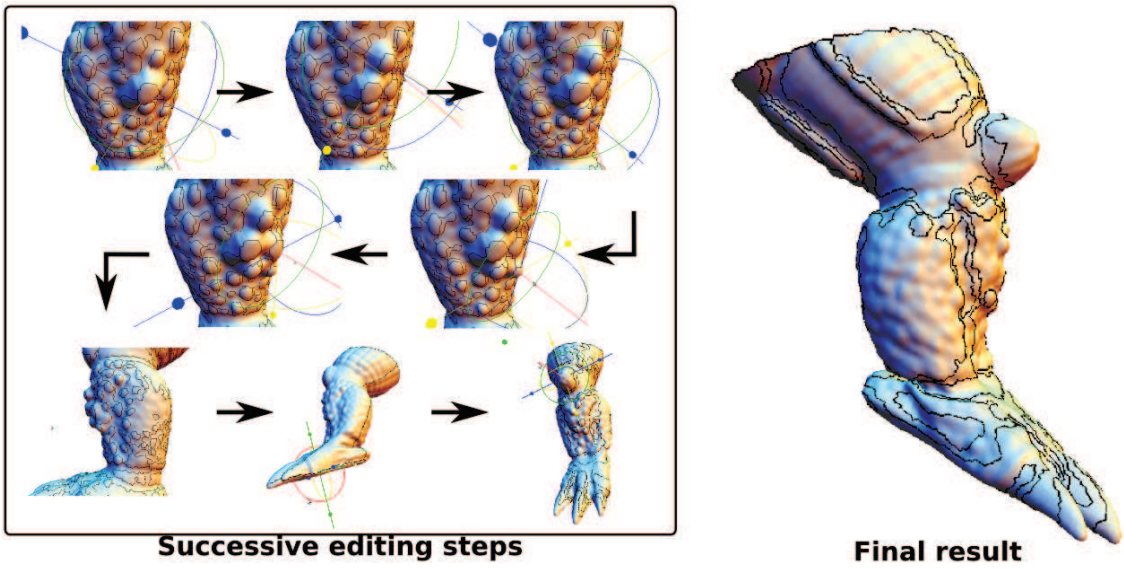
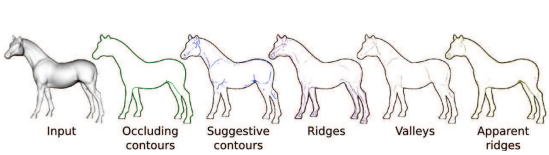


Figure 1.8: Manipulation du complexe de déformation

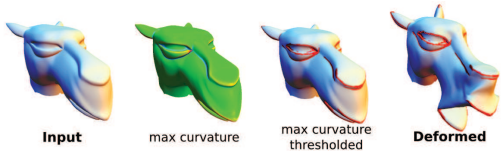
1.3.2 Réseau de lignes pour la déformation

Toutes les parties des objets ne sont pas des caractéristiques pertinentes à attraper et manipuler. Ce constat motive la définition de poignées surfaciques ne dérivant pas nécessairement d'une segmentation de la forme 3D.

Dans cette partie, on présente une interface pour la sélection de *courbes caractéristiques* comme poignées pour la déformation de maillages. On propose de dériver ces courbes des propriétés intrinsèques de la géométrie, comme de propriétés dépendant du point de vue de l'artiste en utilisant des techniques populaires de rendu non-photo-réalistes (NPR), ces deux classes de courbes étant orthogonales et apportant des informations complémentaires sur l'objet considéré.



(a) Courbes issues de techniques de rendu "non-photo-réalistes" (NPR).



(b) Courbes intrinsèques de la géométrie et leur manipulation.

Figure 1.9: (a): Les courbes issues de techniques de rendu NPR décrivent des caractéristiques différentes des objets 3D, et mettent en relief des propriétés difficiles à entrevoir depuis d'autres points de vue. (b): Les propriétés intrinsèques de la géométrie peuvent décrire des structures caractéristiques des objets qui peuvent être complexes à sélectionner manuellement.

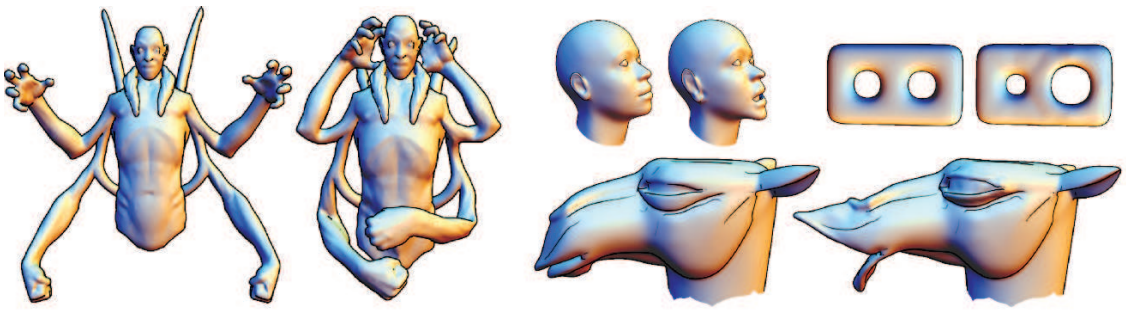


Figure 1.10: Résultats de déformations utilisant des courbes issues de techniques NPR pour la manipulation.

Interface utilisateur Les courbes NPR sont calculées en temps-réel chaque fois que l'affichage est raffraichi à l'écran. Elles sont ensuite régularisées, pour que la forme 3D qu'elle décrive se conforme mieux à l'intuition de l'utilisateur (qui voit des courbes régulières). Cette régularisation est effectuée *par courbe*, à l'aide de l'alternance d'un lissage unidimensionnel et d'une reprojection sur la surface de l'objet à l'aide de technique de projection MLS, ces deux étapes étant répétées plusieurs fois pour obtenir la convergence du lissage.

A chaque fois qu'une courbe est sélectionnée par l'utilisateur, la zone d'influence est composée des sommets proches géodésiquement de la partie de la courbe sélectionnée. L'utilisateur peut régler à tout moment la longueur de la partie de la courbe qu'il sélectionne ainsi que la distance à cette partie définissant la zone d'influence de la déformation. Lorsque l'utilisateur modifie l'objet 3D à l'aide d'une courbe sélectionnée, les propriétés de courbures de la surface sont modifiées, et la courbe servant de poignée de déformation n'a plus de sens. Néanmoins, pour faciliter l'intéraction, il est important que la courbe précédemment calculée et attrapée reste visible pendant le processus de déformation. Les courbes manipulées sont donc maintenues artificiellement, puis supprimées lors de la définition d'une autre poignée de déformation.

Des résultats de déformations d'objets 3D sont présentés en figure 1.10. Ces exemples contiennent des caractéristiques qui sont traditionnellement difficiles à sélectionner et pour lesquelles la technique présentée se révèle parfaitement adaptée (par exemple les lignes de contour).

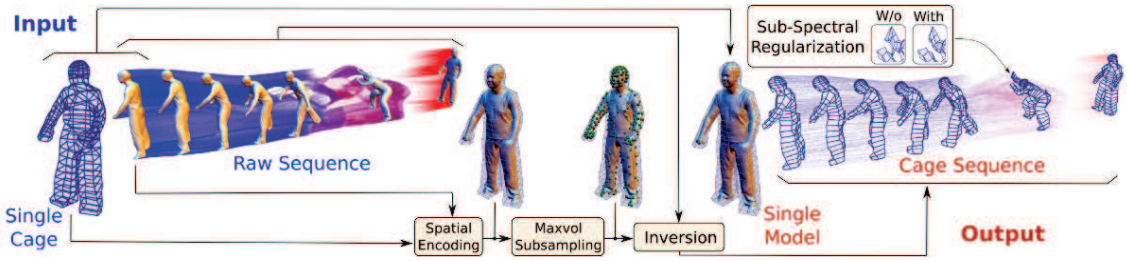


Figure 1.11: **Framework CageR.** De gauche à droite: séquence de maillages animés 3D+t et une cage (entrée), sélection des contraintes pour l'inversion, inversion des différentes poses, régularisation spectrale, création de la séquence de cages animées correspondant à l'animation (sortie).

1.4 Encodage d'animations à l'aide de cages de déformation

Les deux premières parties de cette thèse se sont concentrées sur des objets (i) internes et (ii) surfaciques pour des applications spécifiques de modélisation. Dans cette partie, on considère un ensemble d'objets externes appelés *cages de déformation*, qui permettent de représenter la déformation du volume contenu à l'intérieur de ces cages par la seule donnée de la déformation de la surface de celle-ci. La manière de propager la déformation de la cage à l'ensemble des points de son intérieur s'appuie sur des systèmes de coordonnées barycentriques, qui associe à chaque point de l'espace un influence par rapport à chacun des sommets de la cage. Cela constitue une aire de recherche vaste, et on utilisera dans cette partie plusieurs systèmes de coordonnées, dont le choix revient à l'utilisateur et est considéré comme un paramètre de la technique présentée dans cette troisième partie.

La problématique soulevée ici est la représentation de données animées issues de systèmes de capture haute-définition à l'aide de ces structures bas-niveau. Les avantages de cette méthode sont multiples: (i) la géométrie et le mouvement des données d'entrée sont décorrélées, permettant l'édition indépendante de chacune, (ii) l'animation est compressée, puisqu'au lieu de garder en mémoire un grand nombre de maillages haute-résolution, on ne garde en mémoire qu'un seul d'entre eux, ainsi que la séquence de cages, qui sont typiquement des données de basse complexité, (iii) le système bénéficie directement des propriétés mathématiques des coordonnées de cages, qui ont été développées pour préserver les caractéristiques géométriques des objets dans le contexte de la déformation de ceux-ci, (iv) il est possible de transférer de manière triviale le mouvement de la séquence à d'autres géométries en plaçant un quelconque objet à l'intérieur de la séquence de cages, pourvu que celui-ci soit contenu dans la cage de départ, et (v) il est possible d'utiliser cette représentation de cages comme une représentation haut-niveau de basse complexité pour accélérer significativement les taches complexes de manipulation des données spatio-temporelles, jusqu'à permettre leur exécution en temps réel.

Framework Le framework CageR est illustré en figure 1.11. Prenant en entrée une séquence de maillages animés 3D+t et une cage englobant l'une des poses de l'animation que l'on appelle *pose de référence* (après la phase d'encodage, la géométrie du modèle $\mathcal{M}$ s'exprime en fonction de la géométrie de la cage $\mathcal{C}$ de manière linéaire: $\mathcal{M} = \Phi \cdot \mathcal{C}$, Φ étant

la matrice des coordonnées de cage du modèle), CageR sélectionne de manière optimale l'ensemble des contraintes à prendre en compte pour le problème inverse (étant donnée une déformation du modèle $\mathcal{M}_t$, trouver $\mathcal{C}_t$ telle que $\mathcal{M}_t \simeq \Phi \cdot \mathcal{C}_t$).

Les coordonnées de cage sont inversées à chaque étape t de l'animation à partir de ces contraintes, pour trouver une géométrie de cage correspondant à la déformation. Il est possible d'utiliser un terme de régularité arbitraire sur un sous-ensemble du spectre du problème inverse pour ne l'appliquer que sur les parties les plus instables de la géométrie de la cage ainsi obtenue. La séquence de cages ainsi créée varie de manière régulière.

La complexité des deux stratégies introduites pour le problème inverse (relaxation des contraintes, et régularisation spectrale) se justifie par le fait que les coordonnées de cages populaires (coordonnées aux valeurs moyennes: MVC, coordonnées harmoniques: HC, coordonnées de Green: GC) présentent des conditionnements élevés pour les géométries que l'on considère.

L'étape de relaxation (sélection d'un sous-ensemble de contraintes) permet d'améliorer les propriétés spectrales du système à inverser, et donc de rendre le problème inverse plus stable. L'étape de régularisation spectrale permet d'utiliser des termes de régularisation *destructifs* (tels qu'un Laplacien nul sur la géométrie de la cage) sur la partie instable du spectre seulement, et donc de ne l'appliquer que sur la partie endommagée de la géométrie de la cage.

Sélection des contraintes Pour sélectionner uniquement m points du maillage à prendre en compte dans l'inversion du système de coordonnées (autant de contraintes que d'inconnues), on choisit de sélectionner l'ensemble des m points dont les coordonnées sont les plus stables à l'inversion. Traditionnellement, la mesure de la stabilité à l'inversion d'un système linéaire est donnée par son conditionnement (plus grande valeur propre divisée par la plus petite valeur propre). Chercher une sous-matrice carrée de conditionnement minimal est connu dans le monde de la recherche en algèbre linéaire sous le nom du problème `MinCond`. Ce problème est malheureusement NP-complet, comme ses ϵ-approximations. A la place, on choisit ici une version plus faible du problème, qui donne satisfaction en pratique dans notre cas particulier d'analyse des coordonnées de cages traditionnelles, et qui consiste à trouver la sous-matrice carrée de volume (valeur absolue du déterminant) maximal – problème connu sous le nom de `MaxVol`. La valeur du déterminant est une mesure plus faible de la stabilité d'un système, mais on montre empiriquement que maximiser le volume entraine une diminution du conditionnement (voir chapitre 8). Ce problème est également NP-complet, ainsi que ses ϵ-approximations, mais il existe néanmoins des algorithmes efficaces pour obtenir de manière statistique des solutions acceptables (sans garantie formelle de trouver la solution optimale). Dans la suite, on en présente un que l'on utilise, et qui se révèle être efficace – encore une fois, dans le cas particulier du traitement des coordonnées de cages traditionnelles que nous traitons.

Dans la suite, on note $A_\square$ la sous-matrice carrée composée des premières lignes de la matrice A, et B la matrice construite comme $B = A \cdot A_\square^{-1}$. Une observation simple à établir est qu'échanger les lignes i et j de A, pour lesquelles B_{ij} est l'élément de norme maximale dans la matrice B augmente le volume de $A_\square$ (si $|B_{ij}| > 1$).

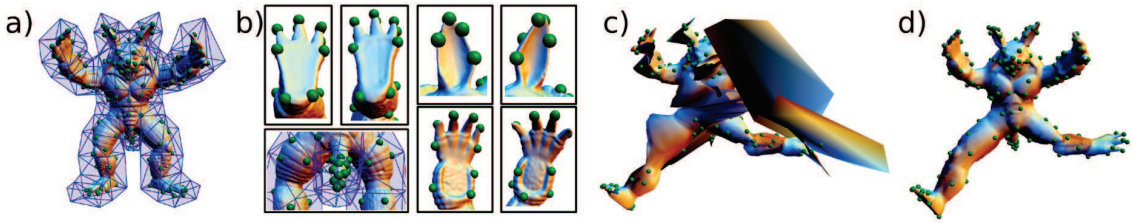


Figure 1.12: **a)** Modèle et cage d'encodage, et contraintes sélectionnées à l'aide de MaxVol (sphères vertes). **b)** Les sommets caractéristiques de la géométrie sont capturés par MaxVol, et la densité locale de la cage se retrouve dans cette sélection (queue de l'animal). **c)** Inversion basée sur des contraintes prises au hasard. **d)** Inversion basée sur les contraintes extraites par notre algorithme.

On note e_k le vecteur colonne avec valeur 1 à l'élement k et 0 partout ailleurs. La mise à jour de la matrice B après cet échange de ligne peut être effectuée sans inverser à nouveau la sous-matrice carrée haute de A, à l'aide de la formule suivante:

$$B := B - (B_{:j} - e_j + e_i) \cdot (B_{i:} - e_j^T)/B_{ij} \tag{1.4}$$

où $B_{:j}$ dénote la j^{me} colonne de B et $B_{i:}$ sa i^{me} ligne.

Un algorithme efficace pour trouver la sous-matrice carrée de volume maximal est donc:

- **Initialisation:** Ordonner les lignes de A, afin que la sous-matrice carrée (m premières lignes de A), notée $A_\square$, soit inversible. Calculer $B = A \cdot A_\square^{-1}$.
- **Itération:** Trouver l'élément de B de valeur absolue maximale B_{ij} (avec $i > m$). Echanger les lignes i et j dans la matrice courrante. Mettre à jour B avec l'équation 1.4.

L'application de cet algorithme algébrique sur les coordonnées de cages (qui sont par définition des objets géométriques), est illustré en figure 1.12. On voit que les caractéristiques géométriques du couplage maillage/cage sont capturées implicitement (figure 1.12 b)).

L'inversion du système pour une pose relativement simple produit de grandes instabilités si les contraintes sont sélectionnées au hasard (figure 1.12 c)), au contraire de lorsque ces contraintes sont issues de la sélection fournie par *MaxVol* (figure 1.12 d)).

Il est aussi à noter que les caractéristiques spectrales d'un système ne sont pas capturées complètement par le conditionnement (μ_1/μ_m, μ_k étant la k^{ime} valeur propre du système), mais aussi par tous les conditionnements d'ordre inférieur (μ_1/μ_k, $k < m$). Ainsi, il est intéressant d'augmenter la dernière valeur propre pour augmenter le conditionnement, mais il est plus pertinent d'essayer d'*égaliser* le spectre entier du système. On illustre en figure 1.13 comment le système issu de la sélection MaxVol améliore l'ensemble du spectre, pour les coordonnées de cages répandues comme les coordonnées aux valeurs moyennes (MVC), coordonnées harmoniques (HC) ou encore les coordonnées de Green (GC).

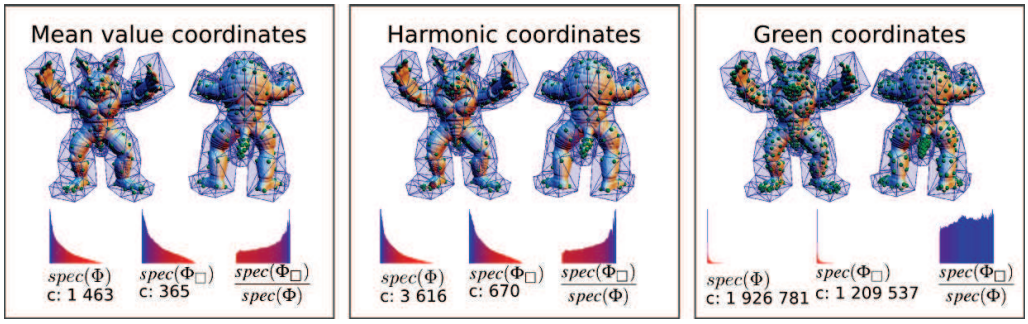


Figure 1.13: Contraintes sélectionnées par MaxVol et spectres des systèmes correspondants pour différents systèmes de coordonnées de cages (MVC, HC, GC). Dans chaque cas, l'utilisation de MaxVol réduit de manière drastique le conditionnement du système (ainsi que les conditionnements d'ordre inférieur) en comparaison du système initial surdéterminé.

Régularisation spectrale On propose ensuite d'appliquer un terme de régularité quelconque (pourvu qu'il soit invariant à la rotation, comme la minimisation du Laplacien de la cage, ou de la norme du Hessien de la fonction de déformation) sur une sous-partie du spectre du système.

Ce choix s'explique par le fait que nombre de termes de régularisation linéaires utilisés courrament (comme un Laplacien nul sur la géométrie de la cage) sont destructeurs, puisqu'ils ne respectent pas la géométrie d'origine du modèle (la cage d'origine n'a pas une géométrie qui est harmonique). Notre stratégie permet donc d'appliquer ces termes de régularisation uniquement sur la partie de la géométrie la plus instable. Des stratégies analogues ont été utilisées pour le filtrage Laplacien sur les surfaces: effectuer un filtrage Laplacien tangentiel revient à n'effectuer que le filtrage sur l'espace tangent à chaque sommet (de dimension 2 au lieu de l'espace d'immersion qui est $\mathbb{R}^3$). Dans notre cas, on choisit de restreindre le filtrage à l'espace vectoriel formé par les derniers vecteurs singuliers associés à la décomposition en valeurs singulières du système.

Formellement, si $U\Sigma V^T$ est la décomposition en valeurs singulières de $\Phi_\square$ (la matrice issue de la sélection MaxVol de la matrice des coordonnées de cages Φ), on peut retrouver au sens des moindres carrés la géométrie de la cage $\mathcal{C}_t$ correspondant à la déformation des contraintes sélectionnées $\mathcal{H}_t$, grâce à la formule: $\mathcal{C}_t = \sum_k V^k \cdot q_{k_t}$, avec $q_{k_t} = (U^k)^T \cdot \mathcal{H}_t / s_k \in \mathbb{R}^3$.

Réaliser un filtrage Laplacien sur les s derniers vecteurs singuliers du système revient à minimiser l'énergie suivante à l'aide d'un système linéaire:

$$E = \lambda || \sum_{k<(m-s)} \Delta \cdot V^k \cdot q_{k_t} + \sum_{k\geq(m-s)} \Delta \cdot V^k \cdot q_{k_t} ||^2 + \quad ||(\Phi_\square \cdot \mathcal{C}_t) - \mathcal{H}_t||^2 \tag{1.5}$$

Les basses fréquences $(q_k, k < m - s)$ sont calculées par

$$q_{k_t} = (U^k)^T \cdot \mathcal{H}_t / s_k, \qquad \forall k < (m - s) \tag{1.6}$$

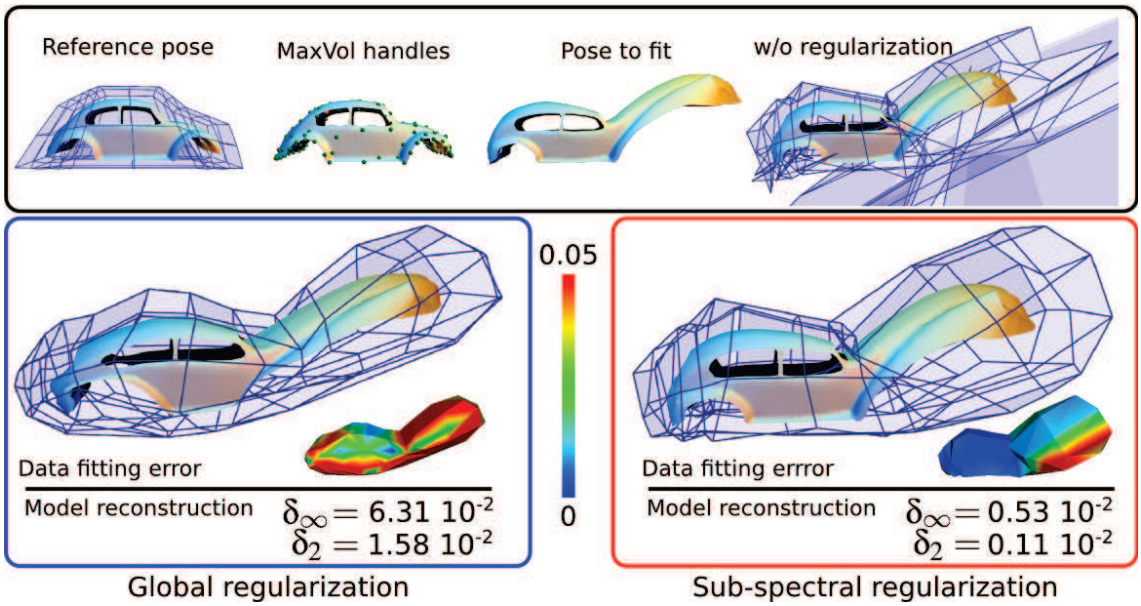


Figure 1.14: Régularisation spectrale (à droite, avec $s = 20$ vecteurs singuliers régularisés), et comparaison avec l'approche traditionnelle (à gauche). Dans les deux cas, on a utilisé les coordonnées aux valeurs moyennes, un terme de Laplacien nul sur la géométrie de la cage comme terme de régularisation, et le facteur de balance d'énergies λ a été fixé à 1.

Calculer les hautes fréquences $(q_k, k \geq m-s)$ de sorte que la régularisation soit effectuée sur celles-ci uniquement revient à inverser le système linéaire suivant (avec comme inconnues $q_{k_t},\ \forall k \geq (m-s)$):

$$\begin{cases} \sqrt{\lambda} \cdot \sum_{k \geq (m-s)} \Delta \cdot V^k \cdot q_{k_t} = -\sqrt{\lambda} \cdot \sum_{k < (m-s)} \Delta \cdot V^k \cdot q_{k_t} \\ s_k \cdot q_{k_t} = s_k \cdot \left((U^k)^T \cdot \mathcal{H}_t / s_k \right), \quad \forall k \geq (m-s) \end{cases} \tag{1.7}$$

Le résultat de cette stratégie de régularisation est illustré en figure 1.14, où on compare les résultats avec une stratégie classique de régularisation sur l'ensemble du spectre du système.

Résultats Quelques résultats sont présentés en figure 1.15 (plus sont disponibles au chapitre 8). La représentation automatique que l'on propose permet de calculer une géométrie plausible des cages pour des séquences animées complexes, telles que celles issues de systèmes de capture haute définition, et contenant des animations de vêtements.

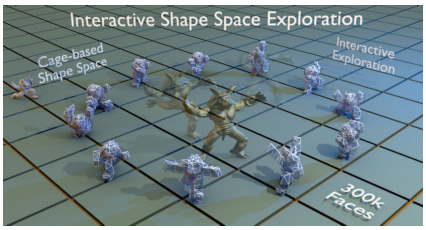
Applications La représentation à base de cages de déformation offre un bon compromis entre compression (figure 1.16(a)) et traitement du signal. En effet, les maillages triangulaires qui décrivent ces objets sont les représentations les plus courantes des surfaces en informatique, et ces maillages sont l'entrée de nombre d'algorithmes de traitement géométriques. On propose comme applications supplémentaires le transfert d'animations à d'autres géométries (figure 1.16(b)), en plaçant n'importe quel objet tenant dans la cage

Figure 1.15: **Séquences reconstruites:** Pour chaque séquence, les poses et cages d'encodage sont montrées, ainsi que les contraintes sélectionnées automatiquement. Pour chaque pas de l'animation, la pose déformée correspondante est montrée, ainsi que la cage sous-jacente et la pose originale à titre de comparaison. Les courbes bleues et rouges montrent respectivement l'évolution de l'erreur moyenne et l'erreur maximale tout au long de l'animation (celles-ci sont exprimées en fonction de la diagonale de la boite englobant la pose d'origine).

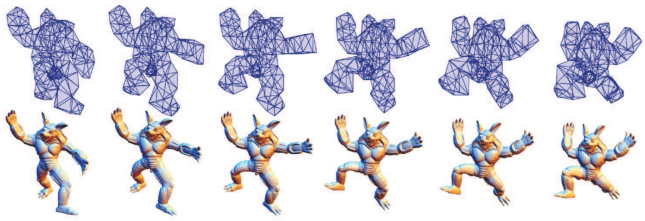
(a) Compression (b) Transfer d'animations

Figure 1.16: (a): Compression d'un maillage animé (499 étapes). (b): En mettant n'importe quel objet dans les cages reconstruites, on peut transférer l'animation à celui-ci.

d'encodage, et la navigation dans l'espace des formes en temps-réel (figure 1.17). Cette dernière application permet de *mélanger* différentes formes compatibles, en effectuant une interpolation barycentrique de celles-ci. Nous avons interpolé les différentes cages correspondant aux différentes poses du même objet (calculées automatiquement à l'aide de notre technique), en utilisant une méthode de l'état de l'art [98]. Ces techniques sont traditionnellement lentes sur les données utilisées ici ($\simeq$ 30 secs. pour une interpolation), alors que l'application requiert de pouvoir parcourir l'espace des formes en temps-réel, car il est difficile voire impossible de pouvoir trouver les différents poids barycentriques à donner à chacune des poses d'origine pour obtenir le résultat souhaité. Notre représentation est adaptée à ce genre de techniques, et permet de l'éxécuter en temps-réel ($\simeq$ 20 interpolations par seconde, visualisation comprise)



(a) Exploration de l'espace des formes



(b) Interpolations le long d'un chemin

Figure 1.17: (a): Exploration de l'espace des formes en temps-réel. (b): Résultats de l'interpolation le long d'un chemin dans l'espace des formes.

1.5 Sur les fonctions spatiales

Dans les parties précédentes, nous avons travaillé sur la construction de *géométries* de nature et de dimension différente, en visant des applications de modélisation ou de manipulation de formes 3D. Dans la troisième partie, nous avons utilisé des cages de déformation pour la représentation de données animées, et la manière dont la déformation était propagée de la cage vers sont intérieur était une donnée du problème.

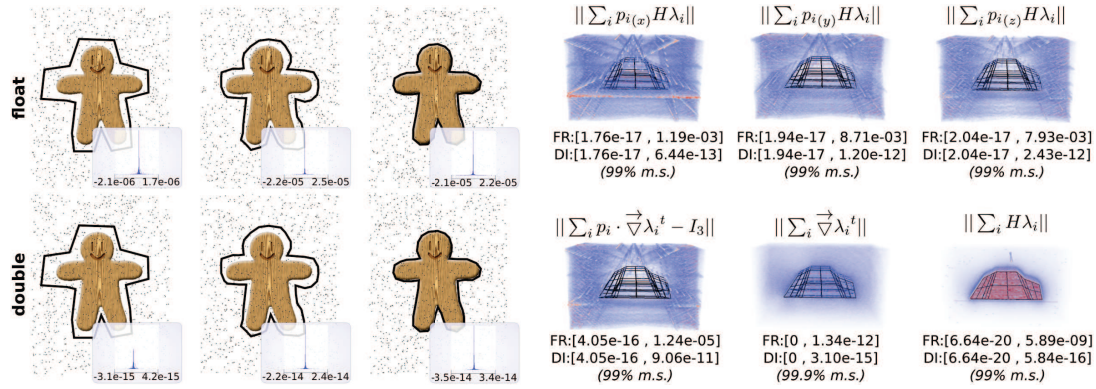
Dans cette dernière partie, nous présentons les résultats mathématiques obtenus sur les coordonnées spatiales, qui définissent une fonction de l'intérieur du volume dans $\mathbb{R}^n$ à partir de contraintes données sur la cage, qui consititue la *frontière* du domaine de définition de la fonction.

Les différentes contributions présentées dans cette partie sont (i) l'obtention des formules analytiques des dérivées des coordonnées aux valeurs moyennes (section 1.5.1) et leurs applications à l'interpolation de fonctions multi-dimensionnelles, et (ii) l'obtention d'une base d'éléments finis permettant la représentation de fonction biharmoniques et leurs applications à la manipulation d'objets 3D (section 1.5.2).

1.5.1 Coordonnées aux valeurs moyennes

Ces coordonnées permettent la définition partout dans l'espace de fonctions scalaires (ou multi-dimensionnelles) f interpolant des valeurs prescrites sur la frontière d'un domaine. Nous nous intéressons au cas de l'interpolation de fonctions définies sur des surfaces triangulaires fermées et orientables par $f(v_i) = f_i$, v_i étant les sommets du maillage et f_i les valeurs de la fonction en ses sommets.

On présente ici très succintement la définition mathématique de ces coordonnées. L'interpolant construit à partir des coordonnées aux valeurs moyennes s'exprime comme une combinaison linéaire de ses valeurs aux sommets, i. e. $f(\eta) = \sum_i \lambda_i(\eta) f_i$, $\lambda_i(\eta)$ étant donc la coordonnée au point η par rapport au sommet v_i du maillage.



(a) 2D (b) 3D

Figure 1.18: **Validation basée sur une solution manufacturée** (groupe de transformations rigides)

Ces coordonnées sont souvent écrites à partir des poids non normalisés $w_i(\eta)$: $\lambda_i(\eta) = \frac{w_i(\eta)}{\sum_j w_j(\eta)}$, avec

$$w_i = \int_{B_\eta(\mathbf{M})} \frac{\phi_i[x]}{|p[x] - \eta|} dS_\eta(x) \tag{1.8}$$

, $B_\eta(\mathbf{M})$ étant la projection du maillage $\mathbf{M}$ sur la sphère unité centrée en η, $\phi_i[x]$ la fonction linéaire par morceaux sur $\mathbf{M}$ qui prend la valeur 1 au sommet v_i et 0 sur tous les autres.

Les coordonnées aux valeurs moyennes ont été utilisées dans le contexte de l'interpolation ("boundary value interpolation"), et de la manipulation d'objets 3D à l'aide de cages de déformation. Dans ce dernier cas, lorsque l'artiste *déforme* la cage, la nouvelle position p_i de ses sommets constitue la fonction à interpoler, et l'objet contenu dans la cage à la phase d'encodage est déformé avec la fonction spatiale $p(\eta) = \sum_i \lambda_i(\eta) p_i$.

Cependant, dans beaucoup d'applications, il peut être désirable de définir des contraintes additionnelles sur les dérivées de la fonction considérer, pour définir la vélocité ou l'orientation du gradient de la fonction par exemple. Dans le contexte de la manipulation d'objets 3D à l'aide de cages de déformation, cela permet de définir des *rotations* en plus de définir des *translations*.

Les calculs nécessaires à une présentation mathématique consistente de ces coordonnées, ainsi que de ses dérivées d'ordre 1 et 2, sont trop longs pour apparaitre dans ce résumé, et nous renvoyons le lecteur au chapitre 10 pour une présentation complète de ceux-ci. Nous nous contenterons de décrire ici la méthodologie développée pour valider numériquement les formules obtenues.

Validation Pour valider empiriquement les formules analytiques obtenues des dérivées, nous nous sommes appuyés sur des solutions manufacturées, une technique populaire en vérification de code [5, 32, 33]. Ce type d'approches de vérification consiste à créer une configuration d'entrée telle que la solution correspondante est connue à priori. L'étape de vérification consiste alors à mesurer à quel point la solution fournie par le programme se conforme à la solution exacte. Il est à noter que cette validation n'est cependant pas

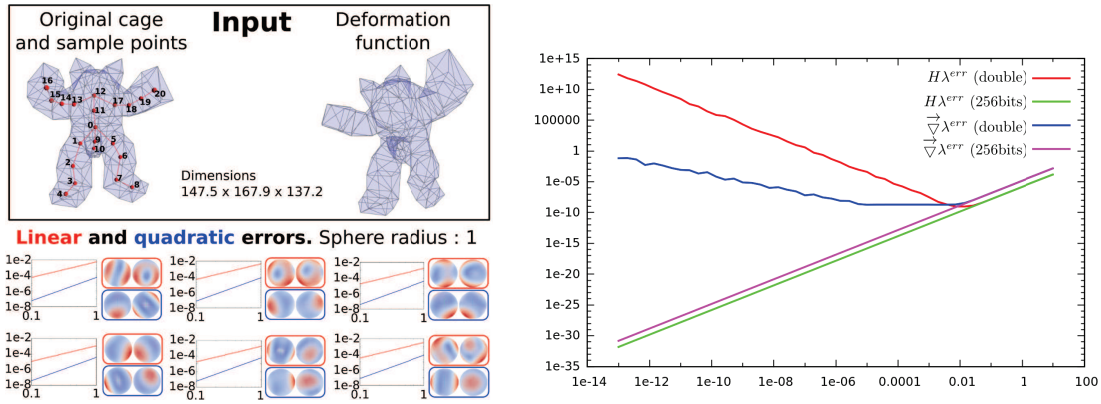


Figure 1.19: **Gauche:** Courbe rouge: approximation linéaire. Courbe bleue: approximation quadratique.
Droite: Validation empirique à l'aide de schémas aux différences finies.

générale, puisqu'elle ne permet que de mesurer les erreurs sur des *instances* de solutions manufacturées.

A partir les propriétés des coordonnées aux valeurs moyennes, nous avons créé une solution manufacturée basée sur les transformations rigides de l'espace, i. e. $\overline{p_i} = T + R \cdot p_i$, T étant un vecteur de translation et R une rotation. Les Jacobiens de la fonction doivent alors vérifier partout $Jf(\eta) = R$ et les Hessiens de chaque coordonnée de f doivent être nuls.

La figure 1.5.1 présente des résultats de calculs de déviations de la solution manufacturée, en 2D et en 3D. On peut noter que les erreurs, bien que relativement faibles en amplitude (relativement à la taille du domaine, et donc de l'amplitude de la fonction interpolée), présentent une amplitude non uniforme, et celles-ci sont généralement plus grandes au niveau des plans support des primitives composant le maillage d'entrée. Cela s'explique par la forme particulière des coordonnées aux valeurs moyennes, qui requièrent pour leur calcul un traitement particulier dans ce dernier cas. On peut noter également que l'amplitude des erreurs sur les gradients et les Hessiens est comparable, voir inférieure, à l'amplitude des erreurs de la fonction elle-même dont le calcul a été introduit par Ju et al. [47].

Approximations linéaire et quadratique Avec les formules des dérivées des coordonnées aux valeurs moyennes en main, nous nous sommes également intéressé au domaine de validité des approximations de Taylor, au premier ordre comme au second ordre. En effet, on sait qu'il est possible de controler l'erreur réalisée par les approximations linéaire et quadratique de Taylor, avec

$$f(\eta + d\eta) = f(\eta) + \overrightarrow{\nabla} f_\eta^t \cdot d\eta + o(||d\eta||)$$

$$f(\eta + d\eta) = f(\eta) + \overrightarrow{\nabla} f_\eta^t \cdot d\eta + \frac{1}{2} d\eta^t \cdot Hf_\eta \cdot d\eta + o(||d\eta||^2)$$

Ces formules ne sont néanmoins valides que dans un voisinage *limite* du point d'intérêt η. Comprendre empiriquement dans quelle mesure ces erreurs sont grandes dans des voisi-

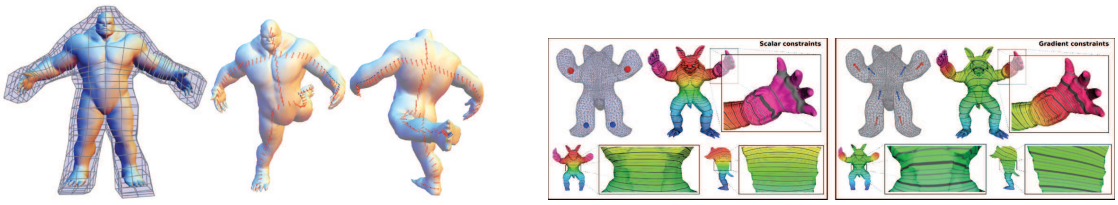


Figure 1.20: **Gauche:** Visualisation des rotations sur le squelette de l'objet. **Droite:** Design de fonctions scalaires utilisant des contraintes sur les gradients pour controler la vélocité du champ scalaire.

nages de taille considérable permet d'appréhender à quel point les contraintes mises sur les dérivées de la fonction seront respectées localement.

La figure 1.19(a) montre les courbes d'erreur de ces approximations (en échelle logarithmique) dans la boule unité centrée en le point d'intérêt (les dimensions du domaîne sont de l'ordre de $150 \times 170 \times 140$). On peut noter que les pentes de ces courbes permettent de vérifier le caractère quadratique de la convergence de l'approximation linéaire et le caractère cubique de l'approximation quadratique.

Nous avons également comparé les formules analytiques obtenues avec des approximations des dérivées utilisant des schémas aux différences finies. La figure 1.19(b) montre la différence entre les deux versions (analytiques et différences finies) pour le domaîne montré en figure 1.19(a). Ces courbes permettent de tirer plusieurs enseignements:

1. les schémas aux différences finies ne permettent pas d'approximer de manière satisfaisante les dérivées des coordonnées aux valeurs moyennes, puisque les valeurs obtenues commencent à diverger pour une taille d'approximation relativement grande avec une précision de type `double` des réels flottants,

2. les schémas aux différences finies semblent converger vers les formules analytiques fournies, lorsque ces erreurs sont calculées avec une précision arbitraire (ici, 256 bits pour la représentation des réels flottants).

Applications La première application utilisant les dérivées des fonctions interpolées à l'aide des coordonnées aux valeurs moyennes est la visualisation: les gradients fournissent l'orientation des iso-valeurs ainsi que la vélocité de la fonction au point considéré, tandis que le Hessien fournit l'information de la rigidité de la fonction (la minimisation de la norme du Hessien de la fonction est d'ailleurs un terme de régularité que l'on rencontre régulièrement dans le domaîne la modélisation géométrique variationnelle). La figure 1.20(a) présente une application de visualisation des rotations de la fonction de déformation sur l'axe médian de l'objet déformé, tandis que la figure 1.20(b) illustre l'utilité de contraindre le champ de gradient de manière locale pour controler la vélocité de la fonction ici créée par l'artiste (les gradients étant des combinaisons linéaires des valeurs aux sommets du domaîne, il est possible de contraindre les gradients à l'aide d'un simple système linéaire).

La figure 1.21 présente un scénario de manipulation directe d'objet 3D, à l'aide d'un solveur variationnel utilisant les coordonnées aux valeurs moyennes.

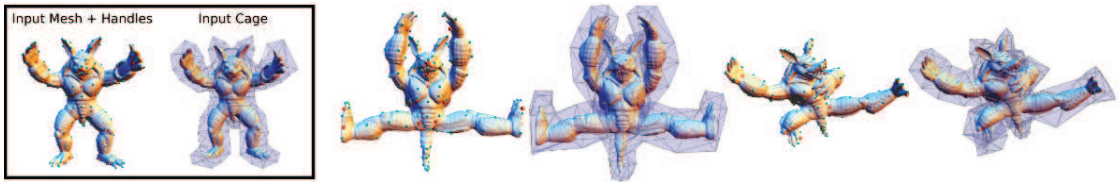


Figure 1.21: Manipulation d'objet utilisant un solveur variationnel et les coordonnées aux valeurs moyennes. Les seules contraintes explicitement données par l'utilisateur sont les contraintes positionnelles (sphères rouges). Les sphères bleues indiquent des contraintes de rotation (de valeur inconnue) sur l'objet.

Puisque notre formulation permet d'exprimer le Jacobien et le Hessien de la transformation partout dans l'espace comme une combinaison linéaire des positions des sommets de la cage, l'utilisateur peut spécifier des contraintes d'orientation (en contraignant la valeur du Jacobien) ou de rigidité (en minimisant la norme du Hessien).

La solution à ce problème d'optimisation est un champ 3D $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, défini partout dans l'espace à l'aide des coordonnées aux valeurs moyennes, qui interpole les valeurs de position des sommets de la cage.

On note $\overline{P}$ l'ensemble des contraintes positionnelles ($\forall v_i \in \overline{P}, f(v_i) = \overline{v_i}$), $\overline{J}$ l'ensemble des contraintes de Jacobien ($\forall v_i \in \overline{J}, Jf(v_i) = \overline{J_i}$), et $\overline{H}$ l'ensemble des contraintes de Hessien ($\forall v_i \in \overline{H}, Hf_x(v_i) = Hf_y(v_i) = Hf_z(v_i) = 0_3$). La solution du problème peut être obtenue en minimisant l'énergie suivante:

$$
\begin{aligned}
E = w^P \sum_{v_i \in \overline{P}} (||\sum_j \lambda_j(v_i) c_j - \overline{v_i}||^2) \\
+ w^J \sum_{v_i \in \overline{J}} (||\sum_j c_j \cdot \overrightarrow{\nabla} {\lambda_j}^t(v_i) - \overline{J_i}||^2) \\
+ w^H \sum_{v_i \in \overline{H}} (||\sum_j H\lambda_j(v_i) \cdot c_{j_{(x)}}||^2) \\
+ w^H \sum_{v_i \in \overline{H}} (||\sum_j H\lambda_j(v_i) \cdot c_{j_{(y)}}||^2) \\
+ w^H \sum_{v_i \in \overline{H}} (||\sum_j H\lambda_j(v_i) \cdot c_{j_{(z)}}||^2)
\end{aligned}
$$

où w^P, w^J, et w^H sont des termes de balance d'énergies pour les contraintes positionnelles, de Jacobien, et de Hessien respectivement.

De manière similaire à [8], la transformation peut être contrainte à être localement une rotation pure. La contrainte de Jacobien s'exprime alors comme:

$$Jf(v_i)^t \cdot Jf(v_i) = I_3 \quad \forall v_i \in \overline{J}$$

Cette optimisation est alors quadratique, mais peut être approximer efficacement à l'aide d'un solveur linéaire itératif.

Discussion Les coordonnées aux valeurs moyennes ont été les premières coordonnées à populariser l'utilisation de cages de déformation sous la définition que l'on leur connait aujourd'hui: des surfaces fermées englobant l'objet à déformer. Elles ont des avantages certains: elles étendent les fonctions à tout l'espace et pas seulement à leur intérieur, ce qui rend plus simple la phase d'encodage d'objets. Elles ont également des inconvénients qui rendent difficile leur utilisation dans certains scénarios. En particulier, elles ne sont pas positives partout, et cela résulte en des distortions de l'objet dans les zones de forte concavité. Le calcul des dérivées des coordonnées aux valeurs moyennes permet de contraindre la transformation de l'objet à être composée de rotations, ce qui – dans une certaine mesure – règle le problème soulevé. Un élément de distinction avec les coordonnées de Green est que les coordonnées aux valeurs moyennes ne requièrent que la valeur des positions des sommets de la cage, et pas les valeurs de normales aux triangles comme c'est le cas pour les coordonnées de Green. Une implication directe de ce point est qu'il est possible, après avoir utilisé le solveur variationnel présenté dans cette section – au contraire des solveurs s'appuyant sur les coordonnées de Green, de reprendre la main sur le controle direct de la cage pour ajouter des détails locaux à la déformation de l'objet après l'avoir mis dans une pose. La session de modélisation peut alors se décomposer en deux étapes distinctes: la mise dans une pose globale de l'objet à l'aide de très peu de points de controle en restreignant l'espace de transformations aux transformations quasi-rigides, puis une étape d'ajouts de détails en manipulant directement la cage optimisée dans la première étape.

1.5.2 Coordonnées biharmoniques

Dans la section précédente, nous avons présenté un outil variationnel pour la manipulation d'objets 3D s'appuyant sur les coordonnées aux valeurs moyennes.

On présente ici une formulation permettant de considérer un espace de fonctions biharmoniques dans ce contexte. Cette formulation s'appuie sur un théorème similaire à la troisième identité de Green: toute fonction biharmonique λ peut s'exprimer à l'intérieur d'un domaîne D en fonction de ses valeurs sur le bord du domaîne ∂D, avec

$$\begin{aligned}\lambda(\eta) = &\int_{\xi\in\partial D} \lambda(\xi)\frac{\partial_1 G(\xi,\eta)}{\partial n_\xi} d\sigma_\xi \\ &- \int_{\xi\in\partial D} G(\xi,\eta)\frac{\partial \lambda(\xi)}{\partial n_\xi} d\sigma_\xi \\ &+ \int_{\xi\in\partial D} \triangle\ \lambda(\xi)\frac{\partial_1 g(\xi,\eta)}{\partial n_\xi} d\sigma_\xi \\ &- \int_{\xi\in\partial D} g(\xi,\eta)\frac{\partial(\triangle\ \lambda)(\xi)}{\partial n_\xi} d\sigma_\xi\end{aligned} \tag{1.9}$$

où g est la solution fondamentale à l'équation biharmonique: $\triangle_1\ g(x,y) = \triangle_2\ g(x,y) = G(x,y)$, et $\triangle_1^2\ g(x,y) = \triangle_2^2\ g(x,y) = \delta_0(||x-y||)$.

On montre dans le chapitre 11 qu'en définissant les conditions au bord d'ordre 0 et 2 comme des fonctions continues, linéaires par morceaux sur la cage, et les conditions au bord d'ordre 1 et 3 comme des fonctions discontinues, constantes sur chaque triangle, on obtient un espace de fonctions biharmoniques qui donne des résultats satisfaisants dans le contexte de la manipulation variationnelle d'objets 3D.

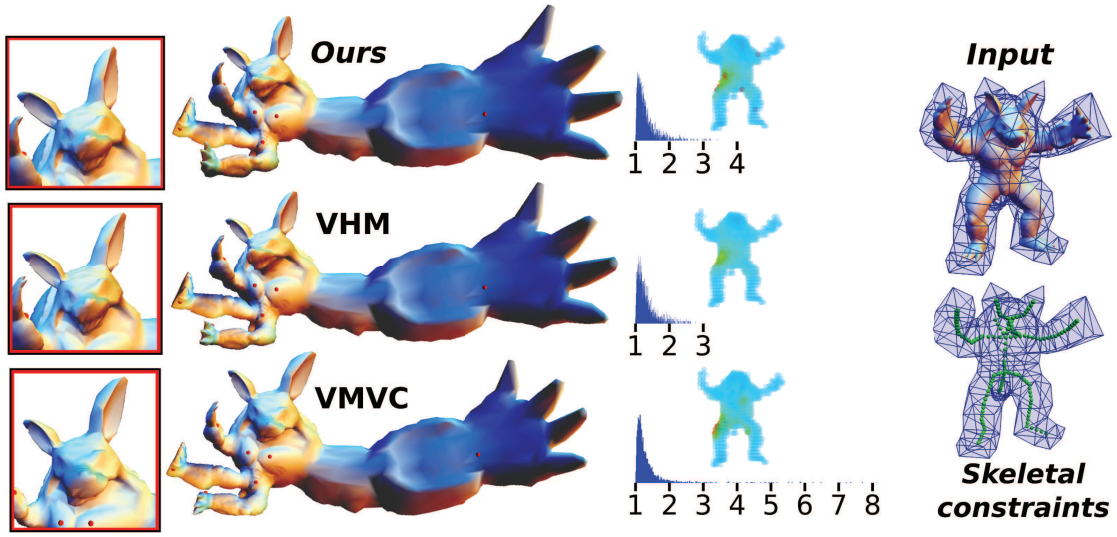


Figure 1.22: Comparaison de notre solveur variationnel biharmonique (VBHM, première ligne) avec VHM basé sur les coordonnées de Green (ligne du milieu) et le solveur variationnel basé sur les coordonnées aux valeurs moyennes présenté dans la section précédente (troisième ligne). Des contraintes de similarité ont été mises sur le squelette de l'objet (sphères vertes).

Ces conditions s'écrivent formellement:

$$\begin{cases} \forall \xi \in \partial D \ : \ f(\xi) = \sum_{v_i} \Gamma_i(\xi) \cdot v_i \\ \forall \xi \in t_j \subset \partial D \ : \ \frac{\partial f}{\partial n_\xi}(\xi) = n(t_j) \\ \forall \xi \in \partial D \ : \ \triangle\,(f)(\xi) = \sum_{v_i} \Gamma_i(\xi) \cdot L_i^V \\ \forall \xi \in t_j \subset \partial D \ : \ \frac{\partial (\triangle f)(\xi)}{\partial n_\xi} = l_j^T \end{cases} \tag{1.10}$$

Les fonctions biharmoniques de déformation ainsi définies s'expriment en fonction de paramètres 3D sur les sommets (positions v_i et valeur de Laplacien L_i^V) et les triangles de la cage (normale $n(t_j)$ et valeur de Laplacien l_j^T):

$$\begin{aligned} f(\eta) = & \sum_{v_i} \phi_i(\eta) \cdot v_i \\ & + \sum_{t_j} \psi_j(\eta) \cdot n(t_j) \\ & + \sum_{v_i} \Lambda_i^V(\eta) \cdot L_i^V \\ & + \sum_{t_j} \lambda_j^T(\eta) \cdot l_j^T \end{aligned} \tag{1.11}$$

Les coordonnées ϕ_i et ψ_j sont *harmoniques*, et leur calcul a déjà été présenté dans la littérature [61], tandis que nous présentons dans le chapitre 11 les formules analytiques des coordonnées *biharmoniques* Λ_i^V et λ_j^T.

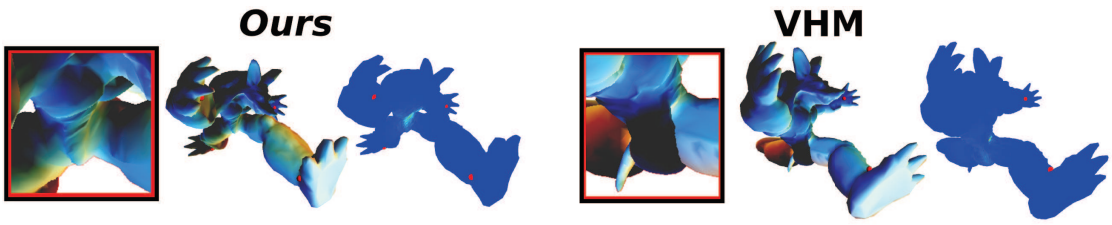


Figure 1.23: Comparaison avec des coordonnées de Green (à droite). Les déformations ont été obtenues avec les mêmes contraintes positionnelles et les mêmes paramètres du solveur variationnel (mêmes énergies minimisées).

Les figures 1.22 et 1.23 présentent des éléments de comparaison avec deux autres types de coordonnées: les coordonnée de Green et les coordonnées aux valeurs moyennes. Il est à noter que nous n'avons pas calculé les formules analytiques des dérivées des coordonnées biharmoniques, et que nous utilisons ici des approximations basées sur des schémas aux différences finies pour celles-ci.

Nous avons également comparé les distortions des objets 3D induites par la manipulation, en se basant sur la définition classique de ce genre de mesure:

$$E_{rot} = \frac{\sum_{t_j \in \mathcal{T}} Aire(t_j)((\sigma_1^j - 1)^2 + (\sigma_2^j - 1)^2)}{\sum_{t_j \in \mathcal{T}} Aire(t_j)}$$

$$E_{sim} = \frac{\sum_{t_j \in \mathcal{T}} Aire(t_j)(\sigma_1^j - \sigma_2^j)^2}{2\sum_{t_j \in \mathcal{T}} Aire(t_j)}$$

, σ_1^j et σ_2^j étant les valeurs propres de la carte linéaire de transformation du triangle t_j [77]. Plus de résultats sont disponibles dans le chapitre 11.

Discussion Nous avons présenté un ensemble de fonctions biharmoniques analytiques dans le contexte de la déformation d'objets basée sur un solveur variationnel. A notre connaissance, les formules analytiques de cette base de fonctions biharmoniques n'a pas été présentée ni utilisée dans la littérature précédemment.

D'autres contextes que la déformation d'objets 3D peuvent également bénéficier de cette base de fonctions, bien qu'il ne serait peut-être pas évident de trouver une métaphore permettant de trouver des valeurs pour les conditions au bord de second et troisième ordres, et des strategies particulières devraient probablement être développées pour chaque cas particulier.

1.6 Conclusion

Dans cette thèse, on a présenté des algorithmes visant à extraire des structures simples qui peuvent aider à la réalisation de calculs complexes sur les objets 3D. Ces structures peuvent être géométriques ou topologiques, peuvent être localisées à l'intérieur de la forme, sur la forme, ou à l'extérieur de la forme 3D. Elles peuvent être des objets de dimension *1*,*2*, ou *3*, ou elles peuvent un simple ensemble d'indices décrivant un sous-ensemble de l'objet.

A partir des contraintes applicatives, nous avons proposé à chaque fois de nouvelles propriétés pour les structures que nous cherchions à construire, que ce soit une définition analytique d'un squelette ou un ensemble minimal de contraintes représentant des déformations de cages. De plus, nous avons introduit de nouveaux traitements géométriques et outils d'analyse permettant l'utilisation de structures classiques d'édition de formes 3D telles que les cages ou les squelettes pour de nouvelles applications.

Sur les objets internes, nous avons proposé un modèle analytique pour les squelettes unidimensionnels, ainsi qu'un algorithme pour trouver une segmentation d'un maillage en disques et cylindres topologiques à partir de laquelle la géométrie du squelette peut être calculée. La construction de cette segmentation est basée sur un algorithme de réduction de graphe, qui utilise un terme de coût de contraction prenant en compte la configuration topologique induite par l'opération pour définir la priorité des opérations ainsi que l'erreur géométrique spécialisée pour la classe topologique considérée. A notre connaissance, ce problème de segmentation de graphe n'avait pas été introduit dans la littérature avant. En pratique, il semble difficile de trouver une stratégie globale permettant de le résoudre, puisque la définition même des erreurs géométriques dépend de la configuration topologique de la segmentation, et donc du résultat en lui-même. Les propriétés particulières de notre modèle de squelette permettent de multiples applications, telles que la définition d'un champ de normales 3D consistent pour le déplacement normal surfacique, et la définition d'un filtre bilatéral préservant les caractéristiques d'ordres de grandeur variés.

Sur les objets surfaciques, nous avons proposé une structure de complexe simplicial, ainsi qu'un réseau de courbes sur la surface, permettant la définition automatique de poignées de déformation. Le complexe de déformation est construit à partir d'une segmentation multi-résolution utilisant des systèmes standard de partitionnement de surfaces, et permet la suggestion de poignées de déformation de types différents (patches, courbes, points) à différents ordres de grandeur. L'interface utilisateur permet de naviguer simplement dans les différents niveaux de résolution de la structure, et d'adapter les stratégies de manipulation à la nature de la poignée de déformation. L'ensemble de courbes que nous avons proposé est construit à partir de propriétés intrinsèques ou dépendant du point de vue de l'utilisateur, et permet d'utiliser des algorithmes standards de rendu non-photo-réaliste pour définir ces poignées linéaires de déformation. Nous avons également introduit une stratégie efficace pour la régularisation de ces courbes et démontré son utilité lors de sessions standards de modélisation.

Sur les objets englobant, nous avons démontré qu'une inversion basé sur une sélection minimale de contraintes minimisant le volume du système associé permettait d'améliorer les propriétés spectrales du système à inverser, et que notre stratégie de sélection couplée à une stratégie de régularisation spectrale étaient adapté au problème particulier de la

représentation d'objets animés à l'aide de cages de déformation. Le retour offert par la solution au problème MaxVol peut également diriger le remaillage local de la cage pour représenter au mieux la séquence animée. Cela ouvre des pistes de recherche intéressantes pour le remaillage efficace de structures de contrôle.

Enfin, nous avons travaillé sur la définition mathématique de coordonnées spatiales, et nous avons obtenu des résultats qui sont de valeur pratique aussi bien que théorique. Nous avons obtenu une formule analytique pour les dérivées des coordonnées aux valeurs moyennes, en 2D et en 3D, et démontré leur utilité pour les applications typiques de ces coordonnées. Nous avons également obtenu une solution analytique pour la modélisation de fonctions de déformations biharmoniques en 3D, et démontré leur pouvoir d'expression dans ce contexte particulier.

Part I

Introduction

Chapter 2

Geometry in Computer Graphics' applications

2.1 Interactive shape editing

Various representations can be used for 3D shapes in Computer Graphics. Polygonal meshes are the most popular one as they are a natural low-resolution representation that can approximate the shape with a controlable error. Physical simulations can be run on these meshes using Finite Elements methods and on-the-fly enrichment of these can be performed very efficiently by applying popular subdivision schemes to them. They also approximate a lot of 3D objects that compose our everyday-world in the most efficient way, such as buildings for example.

2.1.1 Creation

The creation from scratch of a 3D shape is usually separated into several steps, which can be executed by different artists:

- creation of the initial geometry
- remeshing of this geometry for texturing
- detail addition (displacement maps, bump maps,...)
- texturing

These processes are usually performed at different scales, to allow the shape to be represented at different levels of detail. In particular, automatic decimation of 3D objects from a detailed version is a very active field of research and has produced several core contributions in Computer Graphics over the past years. Similarly, high level of details can be generated using subdivision mechanisms, together with displacement functions.

The creation of the initial geometry can be as simple as using a simple cube for the creation of characters for example (we then speak about "box modeling"). More advanced strategies were developed to allow artists to draw simple shapes in order to obtain a complex mesh in a few seconds. For example in ZBrush (Pixologic), an artist can draw a simple skeleton with a sphere on each vertex, these spheres being interpolated linearly on the edges; a quad-dominant mesh can be obtained from this basic geometry, that requires tuning a small set of intuitive parameters; this structure is known as the ZSphere.

2.1.2 Acquisition

3D objects can also be acquired from the real world, by using 3D scanners for example. The output of these algorithms are usually point-clouds, where each point can be attributed with a normal and a color. Low-resolution RGBZ cameras, such as the Kinect, allow to obtain in real-time images containing depth information, and these are used as the direct controller of some video games.

These datasets are generally processed in order to create geometry that can be used by Computer Graphics applications, and often times a triangle mesh is created from these point-clouds. This is especially true when the reconstruction does not need to be performed in real-time, which is the main limitation to the systematic mesh reconstruction from point-clouds today.

2.1.3 Editing

The most straightforward way to interact with such polygonal meshes is to manipulate the vertices it is composed of directly, and eventually edit each polygon locally. Such operations seem to describe an outdated world, but artists have to perform them very often, when they create characters for video games for example. They need to place *extraordinary vertices* carefully, as those create less smooth local geometry when the mesh is subdivided.

Fortunately, high level control is possible. Once a first version of the object is obtained, its shape often needs to be modified to fit particular application needs. Users control these modification processes using interactive systems which allow to explore in real-time the space of possible shapes. There are mainly two classes of modifiers to control the shape of an object. The first is parametric and allows to control the global shape of the object with a very small number of parameters. This control is often decorrelated from the shape itself, and typically exposed with a few numerical values. Such modifiers include twisting, bending and waving operators for instance. The second class of shape modifiers is more object-aware and allows for both local and global shape editing. The key idea with these modifiers is to introduce an intermediate structure which lays out the degrees of freedom offered to the user to edit the shape and acts as a *visual interface* embedded in the 3D space itself and spatially correlates to the shape.

2.2 Structures for shapes

Various high level structures are commonly used in Computer Graphics to control shapes. An important part of those can be classified by dimensionality (even though their geometry is embedded in the 3D space):

- 0 : points, and 3D frames;
- 1 : curve skeletons, that are particularly useful for character animation;
- 2 : surfaces and patches, that are used for variational mesh deformation for example;
- 3 : volumetric structures, such as lattices and cages for deformation.

We can classify the different techniques that transfer the transformations *from these structures to the 3D object*, according to their increasing degree of complexity:

- **rigid:** each point of the object associated with its handle structure is deformed using the same transformation;
- **linear blends:** each point of the object has a transformation that is a linear combination of the transformations of the different parts of the control structure;
- **global coordinates:** each point of the object is expressed w.r.t. the entire control structure and its geometry is directly updated from it;
- **variational:** the position of each point on the surface results from the solution of a large set of constraints which can model the local geometry, the material, user anchors or physics behavior. Depending of the the choosen degrees of constraint, the deformation itself is obtained using a variety of numerical methods, ranging from linear variational ones to high non linear ones.

The automatic or user-assisted creation of these structures, and the way they control the shape, are two distinct, yet closely related, topics of research. The different problematics we tackle in this thesis always refer to either one of these two categories.

2.3 Various geometrical and topological representations

There are plenty of different problems in Computer Graphics, that are difficult to classify or even enumerate (visualization, parameterization, triangular/quadrangular/tetrahedral meshing, animation, segmentation, motion retargetting, shape matching, shape morphing, shape retrieval from data bases, and others). It is natural to say that the structures that are required to solve these many problems are also numerous and various (see Fig. 2.1), and no unique structure allows to solve all problems. These structures are essentially topological and/or geometrical structures (Reeb graphs, spectral analysis, skeletons, surfacic handles, subdivision surfaces, cages, set of points, indices in hash tables, histograms, and others).

In this thesis, we will show that both topological and geometrical structures can be designed specifically for particular classes of shapes and applications at the same time.

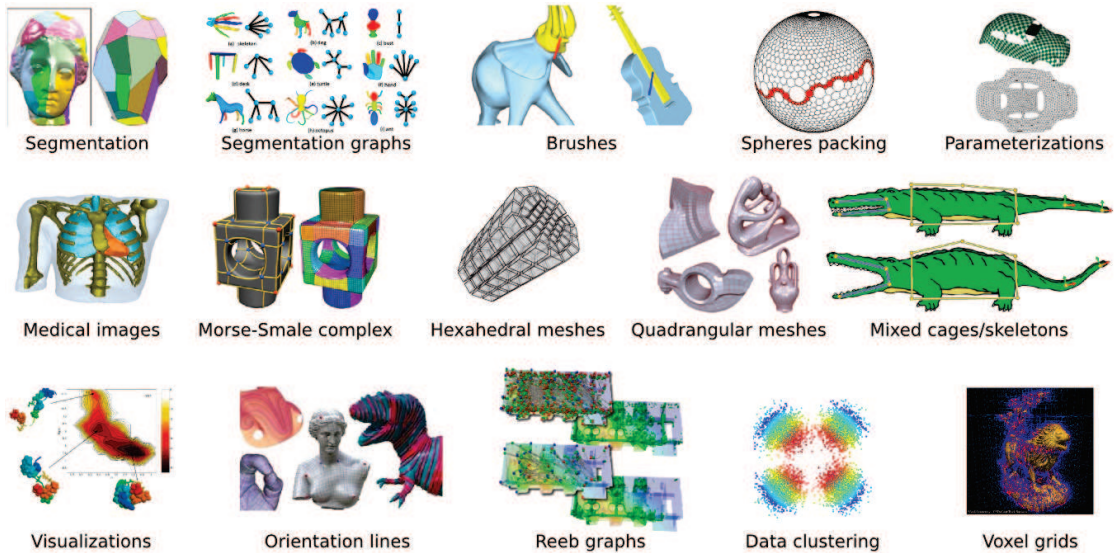


Figure 2.1: Various geometrical and topological structures for specialized processes.

2.4 Shape modeling in the industry

The creation of geometrical content is critical in Computer Graphics and has many applications in various branches of industry. Video games become richer by the day, the environments where they take place become larger, more detailed and more complex.

Some games focus on characters rather than on the geometry of the environment. For CG artists, the strategy that is used has a strong impact: several *tools* are used to create characters, vegetation, or buildings. New problematics appear as well, in particular strategies need to be considered for the creation and optimal representation of different kinds of geometry.

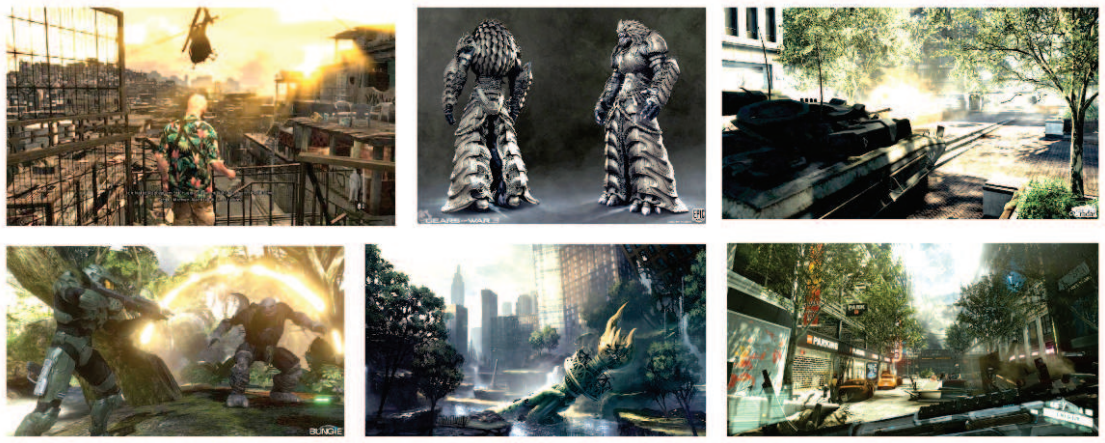


Figure 2.2: Images from recent video games (Crisis 2 - Crytek, Max Payne 3 - Rockstar Games, Halo 3 - Bungie) featuring complex and detailed geometry.

Figure 2.3: Pictures taken from: Pirates of the Caribbean 2, Avatar.

Visual effects and special effects in movies have been improved drastically over the past thirty years. In recent movies (e. g. Avatar, Tranformers) most of the objects that appear on screen are fake, and were created using Computer Graphics tools. The tracking of faces has been studied intensively, in order to map virtual geometry on the actors' faces (see Fig. 2.3).

Computer Aided Design (CAD, see Fig. 2.4) is widely used in industry and has a massive impact on our world. It helps designing and modifying efficiently shapes that fit industrial and technical engineering constraints. It is used intensively for the creation of various objects, ranging from the smallest to the largest, such as electronic devices, phones, cars, planes, or others. It allows to have in the same environment the tools to represent such shapes and tools to run physical simulations on them, such as resistance to heat, crash tests, etc. , and helps designing scenarios to understand the possible consequences of a malfunction of the created object.

Architectural geometry [72] (see Fig. 2.5) is a topic of research that is relatively recent. It poses the problems of fitting precise industrial constraints in the design of architectural structures, such as developability of surfaces (ensuring that those can be obtained by bending flat surfaces), creation of architectural structures with a finite number of primitives (ensuring that a maximum of elements are the same and can be produced at minimum cost), or the exploration of variations of buildings by the manipulation of a minimal set of parameters while guaranteeing consistency.

Figure 2.4: Geometric Modeling in Computer Aided Design.

Figure 2.5: Geometric Modeling in architecture.

2.5 Contributions

We introduce new geometry processing and analysis tools enabling the use of classical shape editing structures such as curve skeletons and cages in new applications. We classify our main contributions as *inner* (part II), *on-surface* (part III) and *outer* (part IV) structures for shape editing. For the latter, we also propose a mathematical analysis of spatial coordinates referring to such structures (part V).

The key concept we develop throughout this thesis is an analysis of the relation between the shape and its structures from the applicative constraints we target. The scientific contributions we make are:

1. an analytic curve skeleton derived from the shape's surface, ensuring a tight connection between both entities that allows for a wide class of signal editing applications;

2. a simplicial complex defined in a multi-resolution fashion and a curves set generated from intrinsic or view-dependent feature lines, allowing easy interaction with the 3D surface at all scales;

3. a set of optimal handles in the context of cage-based representations of animated shapes, together with a new optimization procedure for the regularization of the output cages;

4. a mathematical analysis of popular sets of spatial coordinates, such as mean value and biharmonic coordinates, for which we provide a close formula.

2.6 Outline of the dissertation

In chapter 3 we review some technical background about discrete differential geometry and 3D transformations.

Inner structures

- In chapter 4 we present our analytic model for curve skeletons of surfaces.
- In chapter 5 we present applications that benefit from such a definition, and discuss more generally these applications.

On-Surface structures

- In chapter 6 we present a multi-resolution *simplicial complex* embedded in the surface manifold, allowing the user to grab easily predefined regions as handles for deformation.
- In chapter 7 we present a framework allowing for the definition of curve handles from intrisic and view-dependent curvature for mesh deformation.

Outer structures

- In chapter 8 we present an intuitive framework allowing the conversion of animated 3D meshes into cage-based representations and discuss in detail the *relaxation* as well as *spectral regularization* strategies.
- In chapter 9 we present applications of such a representation for mesh modeling and processing.

Spatial coordinates using cages

- In chapter 10 we present the computation of the derivatives of the Mean Value Coordinates for closed triangular meshes, and illustrate their usefulness for a various set of applications.
- In chapter 11 we present a new set of cage coordinates, that are biharmonic and have a closed-form expression.

Chapter 3

Technical background

Before going into the core of this thesis and speak about the contributions we made, we present some technical background that covers the set of notions that will be useful to the reader.

We are mainly interested in *surfaces* – how we can represent them, model, modify, enrich, and animate them. But, to interact with surfaces, we develop structures that are complex (most of the time, geometrical structures), and those require the understanding of many concepts that are popular in geometry for Computer Graphics.

Throughout this thesis, we will often refer the reader to this chapter, and briefly recall the technical background that is appropriate to each of the specific techniques we will introduce.

The advanced reader to whom discrete differential geometry and space transformations are no secret can skip directly to the next chapter. Others should read this chapter that, we hope, is an as easy as possible introduction to the world of geometry in Computer Graphics.

3.1 Notations

We start this chapter by introducing notations that are going to be employed throughout this thesis.

1. $\cdot \times \cdot$ denotes the cross product between vectors in $\mathbb{R}^3$.
2. $< \cdot | \cdot >$ denotes the dot product between vectors in any dimension. More generally, it is still valid when applied to *tensors*, and $< u|v >= u^i v_i$.
3. n-dimensional vectors $\mathbf{x}$ are noted $(x_1, \cdots, x_n)$, but they are *column vectors* when considering matrix operations (one column, n rows).
4. m_i denotes the i^{th} row of the matrix m, and m_{ij} denotes its element at row i and column j. u_i denotes the i^{th} coordinate of the vector u.

5. $m \cdot u$ denotes the matricial product between m and u.
6. u^t denotes the *transpose* of the matrix (or vector) u. The dot product between two vectors u and v will be then written sometimes $u^t \cdot v$.
7. We note in general $\mathbf{M}$ a 2-manifold surface and $\mathcal{M}$ a triangular mesh with $\mathcal{V} = \{v_0, v_1, \cdots, v_{N-1}\}$ its vertices with associated positions $\{\mathbf{v_0}, \mathbf{v_1}, \cdots, \mathbf{v_{N-1}}\}$, $\mathcal{E}$ its set of edges, and $\mathcal{T}$ its set of triangles.

3.2 Discrete surfaces in $\mathbb{R}^3$

The study of surfaces in $\mathbb{R}^3$ is linked to several mathematical fields, in particular **differential geometry**. It is beyond the scope of this thesis to review all notions of differential geometry. However, we are going to present some of them that are of interest to us. We will present them directly in the context of **discrete differential geometry** where appropriate, where we consider **manifold triangle meshes** as an approximation of a smooth surface.

3.2.1 Manifold meshes

In differential geometry, a set X is said to be manifold of dimension d if and only if for each point η, a small enough neighborhood U_η is homeomorphic to a d-dimensional sphere $V_{\phi(\eta)}$ through a *map* ϕ: $\forall \eta \in X, \exists U_\eta, \phi : U_\eta \to V_{\phi(\eta)} \subset \mathbb{R}^d$. It has boundaries if for some points, their neighborhood is homeomorphic to half a sphere. For surfaces embedded in $\mathbb{R}^3$, the condition is then that the neighborhood of each point is homeomorphic to a disk. This condition translates easily into combinatorial conditions on the set of vertices, edges and triangles that compose the neighborhood of each vertex. In our work, we consider mostly *triangle meshes* as discrete versions of surfaces, but the following definition of *2-manifold mesh* is valid for any polygonal surface embedded in $\mathbb{R}^3$.

A triangle mesh $\mathcal{M}$ is a collection of 3D points $\mathcal{V} = \{v_0, v_1, \cdots, v_{N-1}\}$ called *vertices* and a collection of triangles $\mathcal{T} = \{(t_0^0, t_1^0, t_2^0), \cdots, (t_0^{T-1}, t_1^{T-1}, t_2^{T-1})\}$ that are triplets of integer indices denoting the index in $\mathcal{V}$ of the vertices defining the triangle. The *edges* $\mathcal{E}$ of $\mathcal{M}$ are pairs of vertices belonging to the same triangle. An edge is said to be *boundary* if it has only one adjacent triangle. A surface is said to be *open* if it contains boundaries, *close* otherwise.

We note $V_1(v)$ (resp. $E_1(v)$, resp. $T_1(v)$) the set of vertices (resp. edges, resp. triangles) that are connected to the vertex v, these sets are usually called *one-ring* adjacency sets.

Definition A triangle mesh is *manifold* if:

- any *edge* does not share more than two triangles,
- any *vertex* shares exactly zero or two *boundary edges*,
- any *vertex* has its one-ring-triangles that contains no more than one connected component, when considering edge-adjacency between triangles.

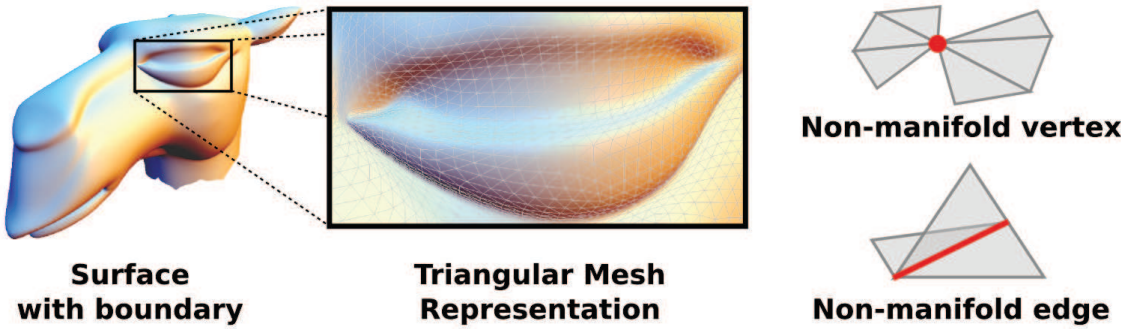


Figure 3.1: A smooth surface can be represented with any precision using a *triangle mesh representation*. The manifold condition (the neighborhood of any point is equivalent to a *disk* or a *half-disk* on boundaries) is easy to transpose to the case of triangles meshes.

The 3D *normal* of a triangle is defined as:

$$n(t_j) = \frac{(v_{t_1^j} - v_{t_0^j}) \times (v_{t_2^j} - v_{t_0^j})}{||(v_{t_1^j} - v_{t_0^j}) \times (v_{t_2^j} - v_{t_0^j})||} \tag{3.1}$$

Definition A triangle mesh is *oriented* if any edge $\{v_i, v_j\}$ has an opposite ordering in its two adjacent triangles t_{ij} and t_{ji} (i. e. $t_{ij} = \{v_i, v_j, k\}$ and $t_{ji} = \{v_j, v_i, l\}$, up to a circular permutation in both triangles).

- A triangle mesh is ordered *Counter-Clock Wise* (CCW) if Eq. 3.1 gives the normal pointing outise for all its triangles.
- A triangle mesh is ordered *Clock Wise* (CW) if Eq. 3.1 gives the normal pointing inside for all its triangles.

It is common to compute *normals* at vertices by averaging the normals of its adjacent triangles, and weighting them by the area of the triangles (see [44]). A more sophisticated solution is to take into account the intersection between the Voronoï area of the vertex and its adjacent triangle as a weighting strategy (see Fig. 3.4), but in our case we chose the prime solution for all our work.

3.2.2 Curvature

The **normal curvature** $\kappa_{\mathbf{n}}$ at a point p with normal n in some direction t in the tangent space is defined as the inverse of the radius of the circle that approximates best the curve defined by the intersection between the surface and the plane spanned by n and t. Equivalently, it is the derivative of the normal in the direction t: $\kappa_{\mathbf{n}}(t) = D_t n$.

For smooth surfaces, it is given in any unit direction (u, v) in a local frame $\{e_1, e_2\} \in \mathbb{R}^3 \times \mathbb{R}^3$ spanning the tangent plane by $\kappa_{\mathbf{n}}(u, v) = (u, v)^t \cdot \mathbf{II} \cdot (u, v)$. $\mathbf{II}$ is a symmetric 2×2-matrix called **the second fundamental form**.

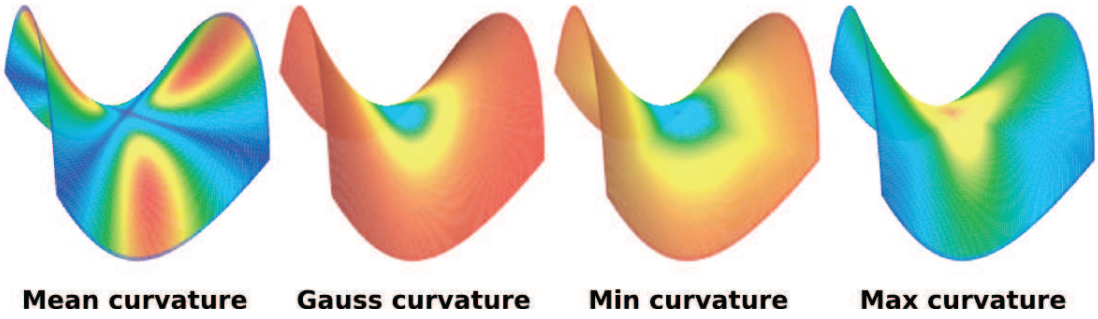


Figure 3.2: Curvature plot of a saddle. Low values are displayed in blue, high values in red. *Image taken from [66].*

Principal curvatures By diagonalizing **II**, we obtain the **minimum** and **maximum curvature** κ_2 and κ_1 as its eigenvalues. They are also called the **principal curvatures.** The associated eigenvectors $\overrightarrow{\kappa_2}$ and $\overrightarrow{\kappa_1}$ are the **principal curvature directions** (it is common to use a 3D vector for their representation, and the principal curvature directions usually refer to $< \overrightarrow{\kappa_2} | \{e_1, e_2\} >$ and $< \overrightarrow{\kappa_1} | \{e_1, e_2\} >$).

Mean and Gauss curvatures The **mean curvature H** is the integral over all directions of the normal curvature $\kappa_\mathbf{n}$.

From the above definition, we can see that $\kappa_\mathbf{n}(\theta) = \cos^2(\theta)\kappa_\mathbf{2} + \sin^2(\theta)\kappa_\mathbf{1}$ in some local frame spanning the tangent plane of the point. By integrating over $\theta \in [0, 2\pi]$, we obtain

$$\mathbf{H} = \frac{\kappa_\mathbf{2} + \kappa_\mathbf{1}}{2} \tag{3.2}$$

The definition of the Gauss curvature **G** is tightly linked to what is sometimes called the *angle deficit* around the point. It is defined as the product of the minimum and maximum curvature:

$$\mathbf{G} = \kappa_\mathbf{2}\kappa_\mathbf{1} \tag{3.3}$$

Computation of curvatures on triangle surfaces There are several works that focus on approximating curvatures on triangle surfaces [16, 66, 76] (patch fitting methods, normal curvature-based methods, tensor averaging methods). We do not review them all in this work, and briefly present the formulation which is used all throughout this thesis, and that was introduced by Rusinkiewicz et al. [76] in 2004.

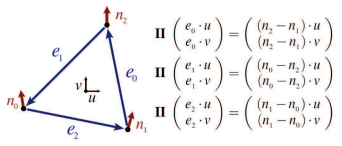


Figure 3.3: **II** on a triangle.

The algorithm works as follows:

1. Since the second fundamental form **II** can be defined in terms of the directional derivatives of the surface normal, it is possible to express it on each triangle in a

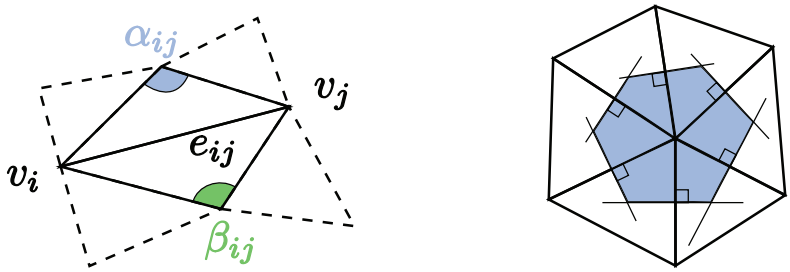


Figure 3.4: Left: cotangent angles of edge e_{ij}. Right: Voronoi area of a vertex.

local frame, in terms of the differential of the vertex normals along the edges of the triangle (see Fig. 3.3 , *image taken from [76]*). A simple system of equations needs to be inverted in order to find **II**.

2. Once **II** is found on each triangle of the mesh, it can be expressed on each vertex of the mesh by averaging the tensors on its adjacent triangles (weighting them by their area). It is required here to rotate the local frame of the triangle in order align best to the local frame of the vertex, so that all tensors are expressed w.r.t. this local frame before being averaged.

3. Once **II** is found on each vertex, it can be diagonalized, and all curvature information can be derived from the expressions introduced in the previous paragraphs (see section 3.2.2).

Note, that this approach allows to compute higher order derivatives, too.

3.2.3 Laplace-Beltrami operator

The **Laplacian operator** $\triangle$, that is a linear operator acting on functions, is particularly important in geometry and in Computer Graphics.

In the Euclidean space $\mathbb{R}^d$, it takes the form $\triangle f = \sum_{i=1..d} \frac{\partial^2 f}{\partial x_i^2}$. On n-dimensional surfaces embedded in $\mathbb{R}^d$ however, it is more challenging to define.

One key observation for the definition of this operator on surfaces (and on triangle meshes in particular), is that it can be decomposed as a combination of the divergence operator $div f = \sum_{i=1..d} \frac{\partial f_i}{\partial x_i}$ and gradient operator $\overrightarrow{\nabla} f = \{\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_d}\}$. Indeed, it is straightforward to see that $\triangle f = div(\overrightarrow{\nabla}(f))$.

Computation of the Laplace-Beltrami operator on triangle surfaces The divergence operator and the gradient operator have been discretized on triangle surfaces, and by using the previous equation, a discretization of the Laplacian operator can be found on triangle surfaces as well. Note, that when considering this definition for the Laplacian operator, it refers to the "Laplace-Beltrami" operator on surfaces.

Note, that it is closely related to the field of *Discrete Exterior Calculus*. For further reference, please see [25].

$$\triangle\,(f)(v_i) = \frac{1}{2A_{Vor}(v_i)} \sum_{v_j \in \mathcal{V}} w_{ij} f_j \tag{3.4}$$

with w_{ij} being called **the cotangent weight** of the edge e_{ij} in case v_j is a neighbor of v_i:

$$\begin{cases} w_{ij} = \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{2} & \forall v_j \in V_1(v_i) \\ w_{ij} = 0 & \forall v_j \notin V_1(v_i), j \neq i \\ w_{ii} = -\sum_{j \neq i} w_{ij} \end{cases} \tag{3.5}$$

The Voronoï set of a point x_i among a collection of points $X_I = \{x_i\}_{i \in I}$ in a space M is the set of points in M that are closer to x_i than to any other point in X_I. This notion is dependent of the distance that is considered on the embedding space M. $A_{Vor}(v_i) = \frac{1}{4} \sum_{v_j \in V_1(v_i)} w_{ij} ||\mathbf{v_i} - \mathbf{v_j}||^2$ denotes the *Voronoï area of the vertex* v_i (see Fig. 3.4); the space that is considered here is the surface manifold, and the distance is the geodesic distance on the surface.

Link to curvature It is pertinent to note that, when applying the Laplacian operator to the coordinates function $\mathbf{x}$(the geometry of the surface), the mean curvature comes as a result:

$$\triangle\,\mathbf{x} = 2\mathbf{H}n \tag{3.6}$$

This equality has motivated previous work regarding the computation of the mean curvature on triangle surfaces, by first computing vertex normals on a mesh, then computing the Laplacian of the mesh vertex coordinates and deriving the mean curvature per vertex as half of the magnitude of the result (by checking for the orientation w.r.t. the vertex normal to set the sign of the mean curvature).

3.2.4 Topological invariants

An important theorem from differential geometry relates the geometry of the surface (through Gaussian curvature) to its topology (through its **genus**):

Gauss-Bonnet For any smooth close surface $\mathbf{M}$

$$\int_{\xi \in \mathbf{M}} \mathbf{G} d\sigma_\xi = 2\pi\chi(\mathbf{M}) \tag{3.7}$$

with $\chi(\mathbf{M})$ the **Euler characteristic** of the surface $\mathbf{M}$.

The Euler characteristic is linked to the **genus** of the shape $\mathbf{g}(\mathbf{M})$ by the following formula:

$$\chi(\mathbf{M}) = 2 - 2\mathbf{g}(\mathbf{M}) \tag{3.8}$$

Figure 3.5: **Region of Influence (ROI)** *Images taken from [13, 14, 63, 86].*

Note, that on triangle surfaces, the Euler characteristic of the surface can be computed from combinatorials, and

$$\chi(\mathbf{M}) = \#\mathcal{V} - \#\mathcal{E} + \#\mathcal{T} \tag{3.9}$$

3.3 3D transformations of surfaces

We present in this section several methods and concepts that are used in the context of surface deformation and surface manipulation. We make a distinction here although the problem is essentially the same: the definition of a function on the surface that represents its modified geometry. The distinction comes mainly from the application side.

In the following, we present **linear variational mesh deformation** techniques (see section 3.3.1) and **space transformation** techniques (see section 3.3.2). Linear variational mesh deformation techniques aim at defining a deformation function directly on the surface, whereas space transformation techniques aim at defining a deformation function on a portion of the 3D space embedding the surface the user wants to deform. Both have their advantages and their drawbacks, and we discuss their respective limitations in section 3.3.3.

3.3.1 Linear variational mesh deformations

We introduce in detail the Linear Rotation Invariant Coordinates (LRI) [63], which is a surface-based method for mesh deformation that we use in some of our work. From the user's point of view, this method belongs to the class of handle-based deformations: at each deformation step, the user needs to define a **region of interest** (ROI) on the mesh with a subset being the **handle** for the deformation – the other part of the ROI acts as a free region, whose geometry is going to be optimized automatically by minimizing some *bending* and *stretching* energies. The remaining part of the mesh is set to be a **fixed region**.

A lot of deformation methods use the same interaction metaphor (see Fig. 3.5), and we refer the reader to [15] for a survey about mesh deformation.

Linear rotation invariant coordinates

The LRI method is based on the definition of **differential coordinates** on the mesh:

- A local frame B_i is *attached* to each vertex v_i (a 3×3-matrix with $n(v_i)$ being the first row, and the second and third rows being two orthogonal vectors spanning the tangent plane of the vertex).
- On each edge e_{ij} a 3×3-matrix T_{ij} is attached, describing the change of the local frames expressed in the local frame B_i: $T_{ij} = (B_j - B_i) \cdot B_i^t$. *This matrix encodes the first part of the differential coordinates of the LRI.*
- Each edge e_{ij} is expressed in the local frame B_i: $d_{ij} = B_i \cdot (\mathbf{v_j} - \mathbf{v_i})$. *This encodes the second part of the differential coordinates of the LRI.*

To deform the model, the user **grabs** the handle, and applies some transformation to it (usually, a simple **manipulator widget** allows to specify a rigid transformation to the whole handle, including a translation, a rotation, and 3 scales that are eventually equal). The algorithm first propagates the rotation and the scales to the entire mesh, thus modifying the local frame of the vertices. In a second step, the position of the vertices are recovered by fitting best the constraints given by the handle and the fixed part, and allows the rest of the mesh to be optimized to fit best the differential coordinates w.r.t. the new frames.

This requires solving two linear systems:

1. **Local frames**: Find $B_i \forall v_i \in \mathcal{V}$, such that $(T_{ij} + I_3) \cdot B_i - B_j = 0_3 \forall e_{ij} \in \mathcal{E}$, the local frames being constrained in the fixed part and the handle;
2. **Vertex positions**: Find $\mathbf{v_i} \forall v_i \in \mathcal{V}$, such that $(\mathbf{v_j} - \mathbf{v_i}) = B_i^t \cdot d_{ij} \forall e_{ij} \in \mathcal{E}$, the position of the vertices being constrained in the fixed part and the handle.

In our implementation, we express the differential coordinates on each edge e_{ij} **twice**, the first time w.r.t. the vertex v_i, the second time w.r.t. the vertex v_j. We noticed that, even though the systems are heavier, the results were significantly improved when symmetrizing the energies. The resulting frames $\{B_i\}_i$ that are solution of the first linear system are not rotation matrices ($B_i^t \cdot B_i \neq I_3$ in general), as this condition is not enforced. After solving the first linear system, we project each matrix B_i on the space of rotation matrices (using SVD) before solving the second linear system. We also weigh each equation in the linear systems by the **cotangent weight** of the edge that is considered.

3.3.2 Space transformations

Although we are mainly interested in deforming *surfaces*, it can happen that we use *Space Transformation* techniques that aim at deforming the entire 3D space (or a portion that is bounded by a closed surface) by the definition of a function $f : \mathbb{R}^d \to \mathbb{R}^d$, and deform the surface manifold M by considering its restriction to it. Popular deformation tools such

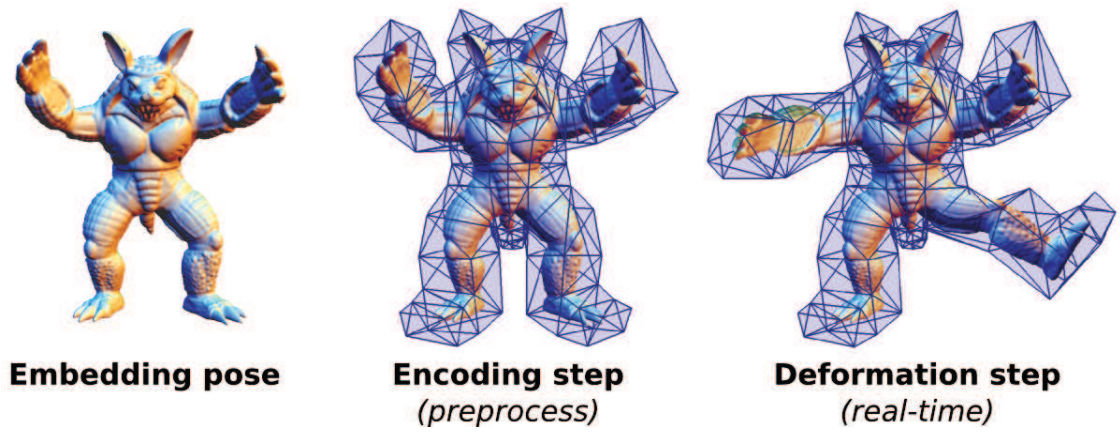


Figure 3.6: **Cage-based deformation pipeline.**

as **control lattices**, **control skeletons** and **control cages** rely on the definition of such functions.

Sederberg et al. [81] presented in 1986 the free-form deformation (FFD) technique, that allowed to use control lattices as a tool for the deformation of its inner space. This lattices had originally the form of a cube, subdivided 3 times in each direction, therefore providing the user with 64 control points. The deformation *inside* the lattice was performed using a Bezier interpolation of the control points of the lattice. Later on, a lot of work has been done on finding more user-friendly primitives for the deformation of shapes using FFD techniques. The control structures evolved to what we call now a *cage*, which is a close triangle mesh surrounding the shape. The definition of the deformation function is then cast to the definition of *generalized barycentric coordinates* w.r.t. the cage vertices.

We refer the reader to the survey of Gain et al. [36] about spatial deformation techniques, and focus in the following on deformations based on cages, as a major part of our work is related to them (see chapters 8,9,10 and 11)..

Deformation cages

Traditionally, the process is composed of two steps. First, the 3D object that the user wants to deform is *encoded* with respect to the cage; this process defines a set of *coordinates* for all points of the object with respect to the cage vertex positions (and the cage triangle normals for Green Coordinates, see paragraph 3.3.2). Second, the user modifies the geometry of the cage and the position of the points of the embedded 3D object are updated with respect to the new geometry of the cage, by multiplying their *coordinates* by the new positions of the cage vertices. This process is illustrated in Fig. 3.6.

In the following, we note ξ a two-dimensional parameter on the cage M, $\Gamma_i(\xi)$ the piecewise linear function that takes the value 1 on vertex v_i and 0 on the others, $d\sigma_\xi$ the surfacic element at ξ, $B_\eta(M)$ the projection of the manifold M onto the unit sphere centered in η and $dS_\eta(\xi)$ its associated surfacic element.

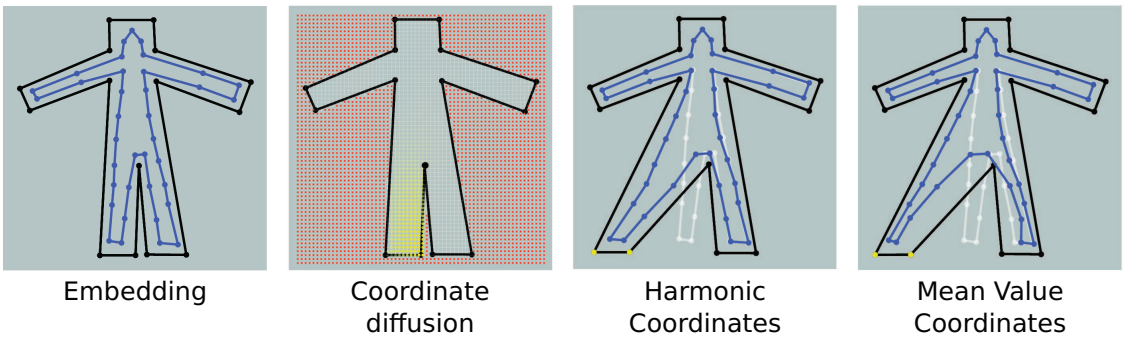


Figure 3.7: Harmonic Coordinates are computed by diffusing each coordinate function on a grid (second). As a result, they are positive. Third and fourth images show a comparison with MVC.

Mean Value Coordinates Mean Value Coordinates (MVC) [47] define an interpolant of a piecewise linear function defined on the cage to the entire 3D space (outside the cage as well). The definition of this interpolant relies on an alternate version of the *Mean Value Theorem* (the function is equals at point η to its averaging over the sphere centered in η, as well as over the ball, for any radius such that the sphere is contained in the domain) that harmonic functions verify.

For a given point η in space, the function is defined as

$$f^{MVC}(\eta) = \frac{\int_{B_\eta(M)} \frac{f(\xi)}{|\xi-\eta|} dS_\eta(\xi)}{\int_{B_\eta(M)} \frac{1}{|\xi-\eta|} dS_\eta(\xi)}$$

Since we can decompose the function f on the manifold M using its associated piecewise linear basis functions – $f(\xi) = \sum_{v_i} \Gamma_i(\xi) f_i$, f_i being the value at the vertex v_i, we obtain that

$$f^{MVC}(\eta) = \sum_{v_i} \lambda_i^{MVC}(\eta) f_i \tag{3.10}$$

with $\lambda_i^{MVC}(\eta) = \frac{\int_{B_\eta(M)} \frac{\Gamma_i(\xi)}{|\xi-\eta|} dS_\eta(\xi)}{\int_{B_\eta(M)} \frac{1}{|\xi-\eta|} dS_\eta(\xi)}$.

MVC define an interpolant of any kind of multidimensional function, but if we set the value f_i to be new 3D positions for the cage vertices, we obtain the deformations induced by MVC.

Harmonic Coordinates Although MVC allow the extension of arbitrary functions to the entire space, they have several drawbacks that make them difficult to use. In particular, they are not *positive* everywhere, and they are not *local*.

Harmonic Coordinates (HC) have been introduced by DeRose et al. [46] in 2006. They are the only harmonic interpolant of the function defined on the boundary of the domain M, and are defined everywhere as the solution of the following system:

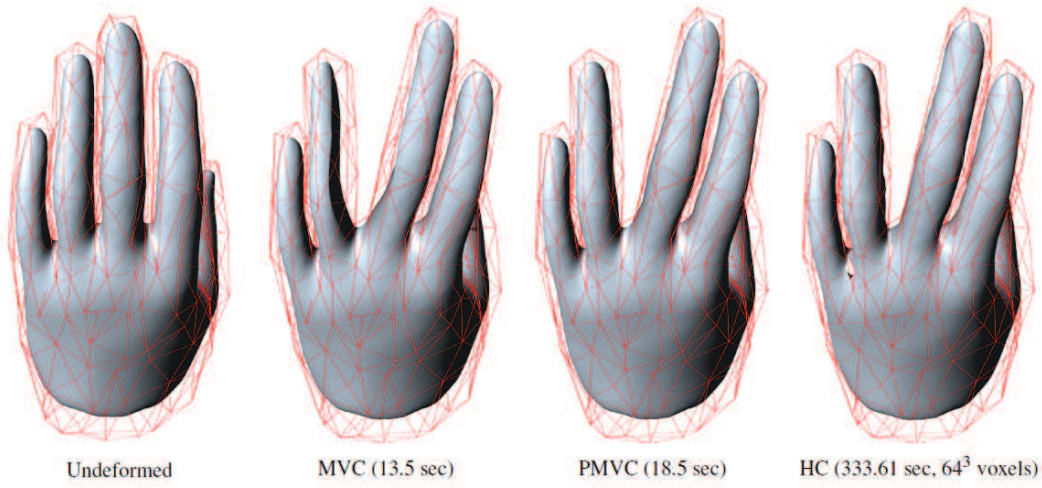


Figure 3.8: Positive Mean Value Coordinates (PMVC, third) achieve similar results as Harmonic Coordinates (HC, fourth) at reasonable rates. In particular, in comparison with Mean Value Coordinates (MVC, second), PMVC are *positive*.

$$\begin{cases} \forall \xi \in \partial D(M): & \lambda_i^{HC}(\xi) = \Gamma_i(\xi) \\ \forall \eta \in D: & \triangle\, \lambda_i^{HC}(\eta) = 0 \end{cases} \tag{3.11}$$

Since no close formula is known for harmonic function defined by Dirichlet conditions on the boundary of the domain, they are approximated by solving this system of equations on a regular grid (see Fig. 3.7).

As a result, these coordinates are *positive everywhere*, *smooth*, and *local*. The main drawback is their computation cost, and the fact that they don't have a close formula make them suited for character animation only (they don't pass the famous "teapot in a stadium" problem).

Positive Mean Value Coordinates Positive Mean Value Coordinates (PMVC) were introduced by Lipman et al. [60] in 2007. These coordinates are defined using the same process as for MVC, but the averaging of the function is performed on the projection of the **visible part of the domain only** on the unit sphere, instead of the whole domain. By noting $VIS_\eta(M)$ the part of the domain M that is *visible* from the point η, the function is given by:

$$f^{PMVC}(\eta) = \frac{\int_{B_\eta(VIS_\eta(M))} \frac{f(\xi)}{|\xi-\eta|} dS_\eta(\xi)}{\int_{B_\eta(VIS_\eta(M))} \frac{1}{|\xi-\eta|} dS_\eta(\xi)}$$

In practice, the evaluation of the *visibility function* VIS is not straightforward, and close formula cannot be derived for the computation of PMVC. Instead, the geometric averaging is performed on the GPU, by drawing basis functions on low resolution *cube maps* centered in each point η of the model.

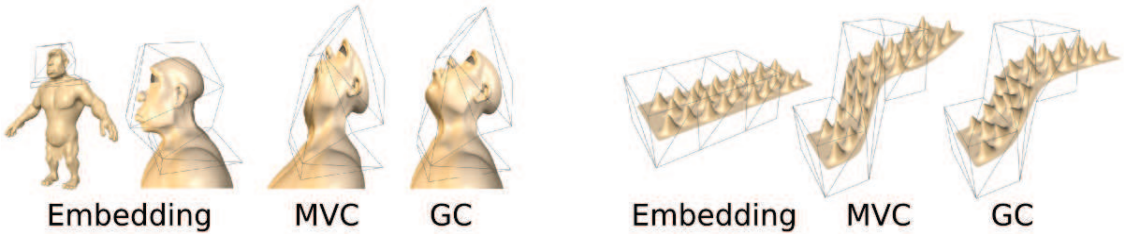


Figure 3.9: Green Coordinates allow for quasi-conformal deformations (left). Rotation on the model can be inferred by translation of the cage vertices (right).

This method allows for the computation of positive coordinates that behave similarly as HC and are equals to MVC for convex domains. However, the computation of PMVC is much faster than the computation of HC (see Fig. 3.8). Note, that the visibility function is not smooth, and therefore this set of coordinates may not be smooth everywhere themselves.

Green Coordinates Green Coordinates (GC) were introduced by Lipman et al. [61] in 2008. Their particularity is that they use in their formulation the normals of the cage triangles. It allows for the first time to induce **rotations** from cage vertex **translations** (see Fig. 3.9 right).

Indeed, for previous cage coordinate systems, where each position η is updated a formula of the form $f(\eta) = \sum_{v_i} \lambda_i(\eta) \cdot v_i$, it is straightforward to see that translating the vertices v_i in a direction $\overrightarrow{t}$ ($v_i := v_i + s_i \cdot \overrightarrow{t}$) results in a translation in the same direction only.

Any harmonic function f inside a close domain D can be written (using Green's third identity) as

$$f(\eta) = \int_{\xi \in \partial D} f(\xi) \frac{\partial_\xi G(\xi, \eta)}{\partial n_\xi} d\sigma_\xi - \int_{\xi \in \partial D} G(\xi, \eta) \frac{\partial f(\xi)}{\partial n_\xi} d\sigma_\xi \tag{3.12}$$

where G is the fundamental solution to the Laplace equation, i. e. $\triangle_x G(x, y) = \triangle_y G(x, y) = \delta_0(||x - y||)$.

Lipman proposed to set the following Dirichlet (w.r.t. f) and Neumann (w.r.t. $\frac{\partial f}{\partial n_\xi}$) conditions on the cage:

$$\begin{cases} \forall \xi \in \partial D : & f(\xi) = \sum_{v_i} \Gamma_i(\xi) \cdot v_i \\ \forall \xi \in t_j \subset \partial D : & \frac{\partial f}{\partial n_\xi}(\xi) = s_{t_j} \cdot n(t_j) \end{cases} \tag{3.13}$$

with t_j being the j^{th} triangle of the cage, $n(t_j)$ being its outward unit normal, and s_{t_j} being the stretch factor associated to the deformation of the cage (its 2D conformality factor).

The resulting deformation induced by the Green Coordinates is expressed as

$$f(\eta) = \sum_{v_i} \phi_i(\eta) \cdot v_i + \sum_{t_j} \psi_j(\eta) s_{t_j} \cdot n(t_j) \tag{3.14}$$

with $\phi_i(\eta) = \int_{\xi \in \partial D} \Gamma_i(\xi) \frac{\partial_\xi G(\xi, \eta)}{\partial n_\xi} d\sigma_\xi$ and $\psi_j(\eta) = -\int_{\xi \in \partial D} G(\xi, \eta) d\sigma_\xi$. These functions are the Green Coordinates, and they are harmonic.

Note, that none of the boundary conditions is fit (a Laplace equation is uniquely determined by a Dirichlet condition **or** a Neumann condition). In particular, if the Dirichlet condition was fit, the result would be the same as the Harmonic Coordinates.

Note also, that the particular choice for this Neumann condition corresponds to setting the Jacobian of the transformation to be as close as possible to a similarity on each triangle of the cage (the Jacobian being determined only in the two dimensional plane corresponding to the triangle where it's considered).

A good mathematical property of these coordinates is that the resulting deformation appears to be *conformal* in 2D and *quasi-conformal* in 3D (see section 3.3.3 for insights on conformality and quasi-conformality).

3.3.3 Comparison and limitations

In this section, we discuss the differences between surface deformations (see section 3.3.1) and space deformations (see section 3.3.2) and present their respective limitations. The notions that are going to be discussed are linked to the field of parameterization, like *conformality* and *quasi-conformality*, that measure the local anistropy of the "stretch" (directional scaling) of the deformed space.

For example, a conformal deformation is going to induce the same stretch in any direction, while the transformation will be locally seen as a rotation and a uniform scaling, therefore it will preserve the visual aspect of textures (as a result, it preserves *angles* of the deformed geometry). This set of transformations are very popular in the field of parameterization, as its primary application is *texturing*.

Conformal space transformations

A function $f : \mathbb{R}^d \to \mathbb{R}^d$ is said to be conformal if and only if its Jacobian matrix Jf is a composition of a rotation and a uniform scale s at each point, i. e. $Jf \cdot Jf^t = Jf^t \cdot Jf = s^2 I_d$.

Finding conformal functions in any dimension has been studied intensively by mathematicians since many decades. It has pratical considerations in the field of Computer Graphics, with applications to *parameterization* for texturing (see Fig. 3.10) or *deformation* of surface meshes.

In two dimensions, several formulations have been proposed to produce conformal deformations [61, 96]. In particular, Green Coordinates are conformal in two dimensions.

In higher dimensions however, conformal transformations $f : \mathbb{R}^d \to \mathbb{R}^d$ are limited to Möbius transformations, which are too rigid to be useful for surface editing. Indeed, Liouville's theorem [58] states that Möbius transformations are a composition of translations, similarities, orthogonal transformations and inversions.

Figure 3.10: Conformal parameterization of a bunny (left) for texturing. Angles are preserved by the parameterization (middle), therefore the texture is not distorted (right). *Images taken from [57].*

Figure 3.11: Spin transformations of surfaces. **Left:** The closest conformal deformation of a twisted bar (right) is the original bar itself (left). Conformal transformations do not allow twist and shear. **Middle:** Conformal deformations of a girafe (left). **Right:** Comparison with Green Coordinates. The color code on the surface indicates the conformality factor of the deformation. *Images taken from [21].*

Conformal surface transformations

Crane et al. [21] presented spin transformations of surfaces embedded in $\mathbb{R}^3$, which allow the definition of conformal deformations when considering them directly on the manifold M (as we have seen, conformal deformations from $\mathbb{R}^3$ to itself are limited to Möbius transformations).

Even though, conformal transformations of surfaces are quite limited as well. For example, *twisting objects induces shear and produces non-conformal deformations* (see Fig. 3.11 left).

We will not go in detail in the presentation of this technique as we do not use it, but it serves the purpose of illustrating the mathematical discussion on surface transformations.

Quasi-conformal space transformations

Quasi-conformal deformations are more general than conformal deformations, and do not offer the exact same guarantees. However, in light of the limitations that are inherent

to conformal deformations, quasi-conformal deformations represent – in our opinion – the best available solution.

A deformation function $f : \mathbb{R}^d \to \mathbb{R}^d$ is said to be quasi-conformal if its Jacobian Jf presents a condition number that is bounded at any point in space. Geometrically, it means that the ratio between the maximal stretch σ_1 and the minimal stretch σ_d is bounded, and therefore the stretch of the deformation is "close" to being independent of the direction in which it is evaluated (a stretch independant of the direction results in *conformal* deformations and preserves angles). See Fig 3.11 (right) for a comparison.

Constant conformality error

A lot of work remains to be done in this field. In particular, a method providing a **constant** conformality error on surfaces or on volumic domains when fitting a user deformation remains to be found. An equivalent in two dimensions has been presented recently by Lipman et al. [59], but their technique does not generalize to arbitrary dimensions.

Part II

Inner structures

In the first part of this thesis, we study *inner* structures and introduce curve skeletons, that are a connected set of curves joining at articulations.

Traditionnally, these structures are used for animation, visualization or shape matching. We show how they can be used on a variety of 3D shapes for shape modeling. We detail the particular constraints this brings, and derive our skeleton model from these observations.

In chapter 4, we present the construction of our skeleton model, which is analytic and derives from the shape through the integration of a piecewise harmonic parameterization on top of a segmentation in topological cylinders and disks.

In chapter 5, we present some applications benefitting from our skeleton model, and discuss its limitations relevant to each of these applications.

Chapter 4

AnaSkel: analytic skeletons

Curve skeletons are usually derived from or embedded near the medial-axis [9] of the shape, and the main constraint for the construction of such structures is, that they are tightly embedded inside the shape. The medial axis of the shape is usually noisy, and not smooth – even for simplified medial axes. This is due to the fact that the distance to the surface is not a smooth function on the medial axis. Because of this, it is difficult to obtain smooth mappings from the surface to the skeleton and vice-versa, making difficult the use of curve skeletons for shape modeling.

In this chapter, we introduce an analytic model for curve skeletons, that is derived from a segmentation of the mesh into topological cylinders and disks. The construction of our skeleton allows to use it as the underlying domain for the decomposition of the geometry of the mesh, and allows for new applications in signal filtering and shape modeling.

4.1 Curve skeletons

Curve skeletons are 1-manifold networks capturing – to a certain extent – the topology and the geometry of a wide class of 3D objects. Their abstraction power has been exploited in various applications requiring analyzing 3D shapes, such as shape matching in visual search [94], shape registration or animation [102]. Basically, they are defined as graphs, with *curved* edges linking 3D nodes and a spatial embedding mimicking the input shape.

Over the many algorithms which have been introduced to compute such skeletons, the primary goal has always been to provide a skeleton with very smooth curves while avoiding any excess of nodes. This stems from the fact that most applications making use of skeletons exploit their coarseness. Unfortunately, in many cases, existing solutions do not model explicitely the relationship existing between a piece of surface and a bone of the skeleton.

We introduce a curve skeleton model, together with its construction algorithm, which embeds such a relationship by the means of a piecewise smooth cylindrical parameterization. Our basic observation is that, up to the desired level of accuracy, a given curve bone should

map to a *topological cylinder* portion of the surface, while extremal nodes should map to *topological disks* of the surface.

This has immediate consequences on the potential applications which can benefit from such a meta-structure. In particular, we demonstrate how shape editing and processing can make use of a *parameteric* skeleton model enriching the 3D surface: for such applications, our skeleton acts as a 1-dimensional domain onto which the surface is expressed. This layout facilitates interactive editing methods such as local shape thickness control or *inset* creation emulating a "geometric peeling" process (e. g. for *Shell Mapping* [71]). Moreover, as the skeleton's embedding reflects faithfully the shape's one, we can express the object's geometry as a signal which is parameterized over the skeleton and can redefine surface processing w.r.t. a "skeletal" basis. In particular, we introduce a new feature preserving mesh filter extending the *bilateral* one [45] to better preserve mid-scale surface structures.

Related work

Extracting a curve skeleton from a given shape is a well studied problem. For a recent overview, we refer the reader to the survey of Cornea et al. [20] and focus on previous methods which are relevant in our context.

Reeb Graphs. A Reeb Graph [35,64,68,74,92] is essentially a data structure for understanding and representing the topology of scalar functions on shapes. Used with euclidean space related functions (e. g. the 'Z' function), it allows to compute a skeleton with a meaningful geometrical embedding by contracting isocontours of such a function into a single point.

In the context of **skeletonization**, the problem is cast to the definition of the proper scalar function whose Reeb graph is going to give the desired skeleton. Note, that this structure is sensitive to noise, but several works exist on the simplification of the underlying function by "at most" ϵ (ϵ-simplification, see [27,28,91] for example) such that the topology is reduced.

Contraction Skeletons. Curve skeletons have also been defined as the result of surface contraction processes. Au et al. [4] use the Laplacian operator on the mesh coordinates to shrink the shape progressively until they obtain a (almost) zero-volume mesh. This contracted mesh is understood as a skeleton and is very smooth. At this point, it is naturally attached to the surface as their is a one-to-one correspondance between the vertices on the surface and the vertices on the skeleton mesh. To get a 1 dimensional skeleton, they collapse edges until they obtain the wanted structure. One vertex of the skeleton then corresponds to a set of vertices of the mesh, which exhibit most of the time a cylindrical shape. However, no parametric relationship is directly available at the end of the process and the pointwise vertex-skeleton relationship lacks control on smoothness and regularity.

Similar approaches [89] have been developed to handle polygon soups and pointsets, allowing to use the skeleton for surface reconstruction and hole filling.

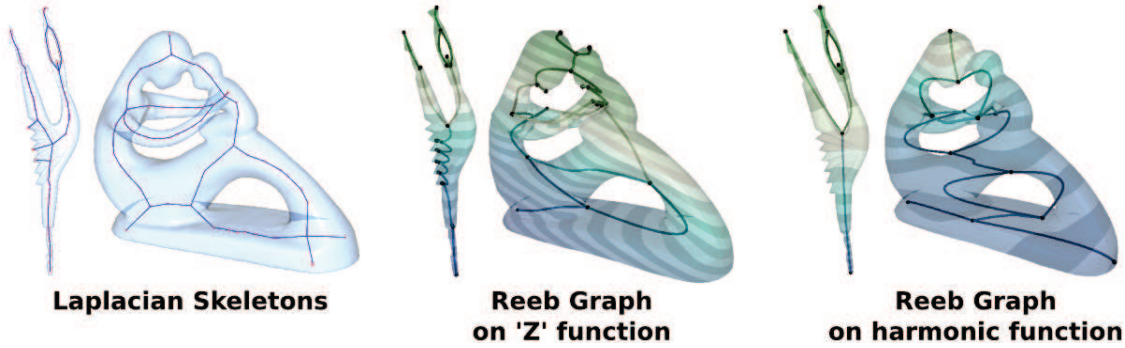


Figure 4.1: **Left: Contraction Skeleton [4]. Middle and Right: Reeb graphs of different scalar fields.**

Alternatively, Sharf et al. [83] cast the problem of curve skeleton generation as a deforming model one, using the midpoint of advancing fronts on the surface to draw the skeleton. Although applicable to non uniform 3D point clouds, this method is sensitive to variations in sampling.

Overview.

While previous approaches are based on Laplacian Contractions or Medial Axis, our method (section 4.2) decomposes the surface in regions which can be parameterized smoothly onto single curves, i. e. *topological cylinders* (section 4.3). In order to guarantee curve bones with regularity inherited from the surface itself (section 4.4.1), we generate the actual bones' geometry by integrating the surface one along harmonic maps (section 4.4.2) automatically constructed from the regions boundaries,with remaining (disk-) regions projecting to skeleton extremities. We also introduce an iterative optimization process inspired by Centroidal Voronoï Tessellation (section 4.5) which improves the surface decomposition. Compared to other curve skeletons (section 4.6), the regularity and the parametric nature of the skeleton model makes it suitable for interactive modeling and processing of shapes (chapter 5): we demonstrate its use for modeling applications with shape editing and *inset surface* design tools, as well as the benefit it brings to geometry processing with a *skeletal* feature-sensitive mesh smoothing operator.

Contributions. The main contributions of this technique are:

- a curve skeleton model with an analytical form and the following properties:

  Smoothness: each bone is a smooth 1D-curve;

  Harmonicity: each bone embeds a harmonic map with a topological cylinder on the surface;

  Regularity: each bone's geometry corresponds to its surface region, where *trajectories* (i. e. , surface-skeleton vector field) have good differential properties;

- an efficient algorithm to construct it from a surface mesh;

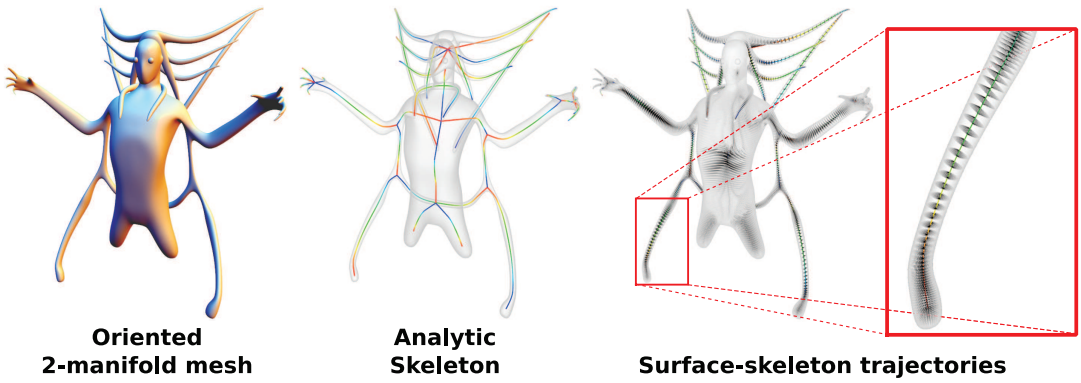


Figure 4.2: **Analytic Curve Skeleton** of a polygonal surface mesh. Smoothly varying surface-skeleton trajectories offer a versatile layout for shape editing and processing.

- a disk-cylinder surface decomposition structuring harmonic maps over the surface which encodes the surface-skeleton correspondances;
- a closed-form geometric embedding of the skeleton, with regularity derived from the surface.

We also illustrate skeletal modeling and processing with three practical contributions:

- an interactive shape modeling tool allowing to edit shape thickness locally;
- an inset surface modeling tool exploiting the surface-skeleton inter space with application to Shell Mapping;
- a feature-sensitive mesh filter based on skeleton-aware shape thickness.

Geometric cylinders decomposition. As mentioned earlier, curve bones can be interpreted as central structures of *topological* cylinders. Finding *geometric* cylinders inside 3D data [6, 10, 17, 73] has been studied to discover semantic shape information from raw geometry such as 3D point clouds. In this context, geometry-based segmentation techniques, [49, 50, 75, 99] are very efficient at finding cylinder-like clusters. Unfortunately, as we are interested in **topological** cylinders – with geometry often differing significantly from conventional cylinders – we have to define a new algorithm with cluster topology as the major constraint.

Topology-based decomposition. Topological disk surface decomposition is a classical process in surface parameterization [3,57,77,78,85]. The idea is to segment the surface into several disks, with the geometry of the disk controlled by an error function. The topology of the decomposition can be assessed by computing the Euler characteristic and inspecting the adjacency configuration of the region boundaries [26, 29]. In our context, we seek for regions which are homotopy equivalent to cylinders, which is a different problem than

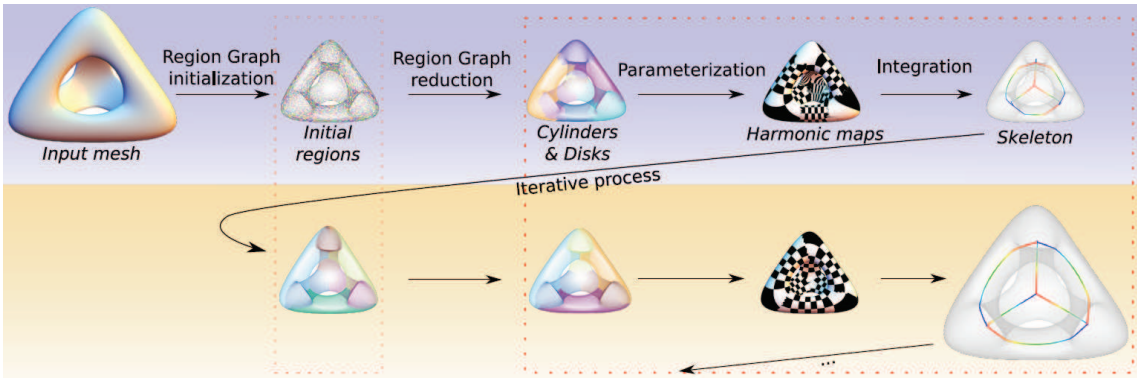


Figure 4.3: **Algorithm overview:** the skeleton creation process (top), followed by optional optimization (bottom).

finding geometric cylinders to approximate the shape. While the latter can effectively be addressed with variational methods [19,99,101], we propose a new approach, suitable to find topological cylinders and which takes inspiration from both adaptive surface optimization methods [40] and hierarchical segmentation [82].

4.2 Our algorithm at a glance

Given a triangulated oriented 2-manifold mesh, our algorithm outputs a skeleton composed of *articulations* (3D points) and curve bones linking them, together with a disk-cylinder surface decomposition and harmonic maps defining the relationship between both:

$$\text{Input mesh} \longmapsto \begin{cases} \text{Analytic Skeleton} \\ \text{Cylinders and Disks Decomposition} \\ (b, u, \theta)\text{-Parameterization} \end{cases}$$

Our algorithm starts by defining an initial segmentation made up of topological disks and applies 3 processes (Fig. 4.3):

1. the dual graph of the segmentation is reduced so that the different regions merge together and create "well-shaped" cylinders;
2. an harmonic parameterization is computed for each region;
3. a bone is generated from each harmonic map.

4.3 Cylinder-disk segmentation

Any 2-manifold can be tessellated with an infinite number of topological disks. In order to find a reduced set of large cylinders and disks, we adopt a bottom-up strategy where we start from many regions and progressively aggregate them to form cylinders and larger disks.

4.3.1 Bottom-up strategy

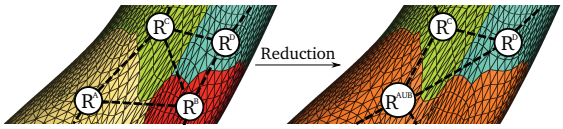


Figure 4.4: **Region graph reduction**: collapsing an edge (black dot lines) is equivalent to merging two regions.

The segmentation is initialized by defining a large number of small regions on the surface. These regions are connected sets of triangles and the so-defined clustering must satisfy the *closed-ball property* [29], i. e. each region is homotopy equivalent to a disk and the (non-empty) intersection between 2 partitions is a closed 1-ball, i. e. a single open curve [26]. For the sake of simplicity, the algorithm can be safely initialized with one triangle per region to guarantee this property. Alternatively, a space subdivision structure (e. g. octree) can be adaptively refined until meeting such a condition in each leaf, with leaf triangle sets becoming initial regions.

The region adjacency graph (or *Region Graph*) is defined as a set of nodes (one for each initial region) and a set of edges (one for each pair of regions sharing at least one triangle edge). The region graph provides a single *reduction* operator by the means of successive edge collapse merging 2 nodes/regions and removing one edge of this graph (see Fig. 4.4).

Once the *region graph* is initialized, all possible edge collapses are pushed into a priority queue, with the priority defined for the collapse of edge $\{A, B\}$ as a 2-term value $\{e_T(A,B), e_G(A,B)\}$:

- e_T is defined on a discrete scale of *topological classes* and models the topological change induced by the collapse,
- e_G is defined on a continuous scale and models the geometric cost of the collapse.

Edge collapses are ordered w.r.t. increasing topological cost e_T, and then w.r.t. increasing geometrical cost e_G when two candidates have equal e_T.

When processing an edge-collapse, the two nodes of the edge are merged and their neighborhood in the graph is updated. Last, the two nodes are flagged as *merged* so that they cannot be merged later in the queue.

At this step, new candidate edge collapses can be pushed into the queue if they pass a *discarding test*, which depends on both, topology and geometry (see section 4.3.4).

4.3.2 Topological priority term

The topological priority term e_T of a candidate collapse of edge $\{A, B\}$ depends only on the topology of the region sets $\{R^A, R^B, R^A \cup R^B\}$. We summarize the different cases in Table 4.1.

4.3.3 Geometric priority term

In order to compute the geometric cost of a collapse, we use *proxies* [19] to capture the shape of regions. A proxy P^A is a triplet $(\mathbf{p}^A, \mathbf{n}^A, s^A)$ representing a surface region R^A, with $\mathbf{p}^A$ its barycenter, $\mathbf{n}^A$ its average normal and s^A its area. The collapse operator requires to build a new proxy for the emerging region. When $R^X = \bigcup_i t^i$ is the union of triangles t^i with barycenters p_i, normals n_i and areas s_i, then $s^X = \sum_i s_i$, $p^X = \sum_i s_i \cdot p_i / s^X$, and $n^X = \sum_i s_i \cdot n_i / s^X$. The expressions of p^X and n^X correspond to the natural discretization of the averaging of the functions p and n defined by surfacic integration on X. Therefore, if $R^X = \bigcup_i R^i$, we obtain P^X by additivity on $\{P^i\}$.

Let $\langle \cdot | \cdot \rangle$ (resp. $\langle \cdot \bigotimes \cdot \rangle$) be the dot (resp. cross) product between two vectors. For each topological priority term k, we define a specific geometric cost function E_k ordering candidate collapses which are topologically equivalent and affect $e_G(A,B) = E_{e_T(A,B)}(A,B)$. Let $R^X = R^A \cup R^B$. The definition of E_0 must favor hole filling while penalizing emerging flat "caps" at cylinder's extremities:

$$E_0(A,B) = 1 - \|\mathbf{n}^X\| \tag{4.1}$$

E_1 must favor the creation of cylinders while E_4 should detect the possible union of cylinders leading to valid ones, we model both errors using a low mean normal:

$$E_1(A,B) = E_4(A,B) = \|\mathbf{n}^X\| \tag{4.2}$$

At this point, we do not aim at defining an orientation for cylindrical regions. Instead we only exploit the fact that for ideal cylinders, the average of all normals is the null vector (which is equivalent to minimizing E_1).

Geometric errors E_2 (*Merge Disks*) and E_3 (*Grow Cylinder*) are defined as regions' standard deviation of which the minimum favors sphere-like shapes:

$$E_2(A,B) = E_3(A,B) = \sum_i s_i \cdot \|\mathbf{p}_i - \mathbf{p}^X\|^2 / s^X \tag{4.3}$$

However, without normals, this error is blind to orientation and trades concavity for convexity. As a consequence, it cannot be used alone to drive the algorithm, but gives satisfying

Topology			**Name**	e_T
R^A	R^B	$R^A \cup R^B$		
Cyl.	Disk	Disk	*Fill Hole*	**0**
Disk	Disk	Cyl.	*Create Cylinder*	**1**
Disk	Disk	Disk	*Merge Disks*	**2**
Cyl.	Disk	Cyl.	*Grow Cylinder*	**3**
Cyl.	Cyl.	Cyl.	*Merge Cylinders*	**4**
Other cases			*Skip*	∞

Table 4.1: Topological priority term as a function of the topological change. Candidate edge collapses are ordered in the priority queue by increasing topological priority term.

results in conjunction with the validity predicate, in particular the *Quasi-Star* estimator (see below).

4.3.4 Validity predicate

When an edge is collapsed, two regions merge and new potential collapses occur. However, a large number of them lead to worse configurations with complicated topology and/or bad geometry. We prevent such degeneration using a *Validity Predicate* $\mathcal{V}$ indicating if an edge $\{A, B\}$ can be inserted in the priority queue. The topological part of this predicate has been mentioned before (see Table 4.1): any edge corresponding to a collapse with an infinite topological error e_T is ignored.

$$\text{(i)} \quad e_T(A,B) \notin [0,4] \Rightarrow \mathcal{V}(A,B) = false \tag{4.4}$$

The geometrical part of $\mathcal{V}$ checks e_G to prevent segmentation discrepancy. In particular, "ideal caps" should be preserved at cylinders' extremity. These cases being equivalent to a semi-sphere in $\mathbb{R}^3$, the mean normal over them should equal 1/2 when a "Fill Hole" edge-collapse is considered:

$$\text{(ii)} \quad \left\{ \begin{array}{l} e_T(A,B) = 0 \\ E_0(A,B) < 1/2 \end{array} \right. \Rightarrow \mathcal{V}(A,B) = false \tag{4.5}$$

Note that 1/2 is **exactly** the length of the averaged outward normal of any semi-sphere (i.e. any positive radius).

Classical *meaningful decomposition* metrics [55] are too restrictive for generalized cylinders, which may not be convex. Instead, we take into account orientation and penalize tunnels to be interpreted as cylinders by defining a *Quasi-Star* estimator $\mathcal{Q}$ on the union of two regions:

$$\mathcal{Q}(R^X) = \sum_i s_i \cdot \langle \mathbf{p}_i - \mathbf{p}^X | \mathbf{n}_i \rangle \tag{4.6}$$

and add a third test to our predicate :

$$\text{(iii)} \quad \left\{ \begin{array}{l} e_T(A,B) \notin \{3,4\} \\ \mathcal{Q}(R^A \cup R^B) < 0 \end{array} \right. \Rightarrow \mathcal{V}(A,B) = false \tag{4.7}$$

Since we seek for *generalized* cylinders, we do not use the Quasi-Star estimator when a "Grow Cylinder" or a "Merge Cylinders" edge-collapse occurs.

If the three tests fail, then $\mathcal{V}$ is true and $\{A, B\}$ is inserted into the priority queue.

4.3.5 Segmentation results

In practice the entire process can be sped up when starting from a coarse initial segmentation (e.g., octree partition). Fig. 4.5 shows several examples of surface decomposition with our algorithm.

Figure 4.5: *Region Graph* construction results.

4.4 Skeletonization

With a disk-cylinder surface decomposition in hand, we can now generate the curve bones. Each cylinder is contracted to a smooth 1-curve and each disk (which ideally corresponds to a cylinder "cap") is contracted to a single point, located at the extremity of a bone. We propose to conduct this process using an harmonic map on the regions.

4.4.1 Harmonic parameterization

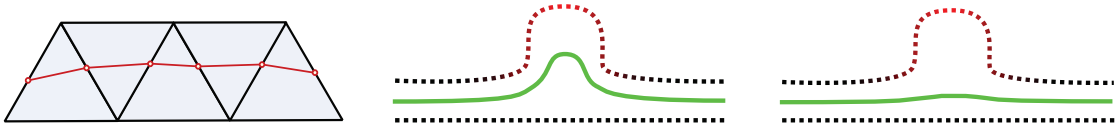
We note $\triangle$ the Laplace operator onto the manifold surface. We use the approximation of [69] to solve the linear systems 4.8, 4.9 and 4.11 in a least-squares sense onto the mesh vertices. We use the CHOLMOD library for that purpose.

(u, θ) **cylinder parameterization.** Considering Fig. 4.6, the u-parameterization of a cylinder is obtained by solving the linear system:

$$\begin{cases} u(v) = 0 & \forall v \in U_0 \\ u(v) = 1 & \forall v \in U_1 \\ \triangle u(v) = 0 & \forall v \notin U_0 \cup U_1 \end{cases} \tag{4.8}$$

The θ-parameterization is obtained by first cutting the cylinder with a path that follows the best $\overrightarrow{\nabla} u$, and then solving a similar linear system to Eq. 4.8:

$$\begin{cases} \theta(v) = 0 & \forall v \in \theta_0 \\ \theta(v) = 2\pi & \forall v \in \theta_{2\pi} \\ \triangle \theta(v) = 0 & \forall v \notin \theta_0 \cup \theta_{2\pi} \end{cases} \tag{4.9}$$



(a) Iso-Curve (b) Spatial Weight (c) θ-Dist. Weight

Figure 4.7: A cylinder (dot line), its θ-distortion (dot line color) together with its bone (green line). The integration of the u-iso-value (right, Eq. 4.12) yields almost no noise.

To obtain the cut, a cost function is attributed to each edge $\mathbf{e}$ of the cylinder that calculates the deviation between the orientation of the edge and the normalized gradient of u $\vec{\nabla}_1 u = \vec{\nabla} u / \|\vec{\nabla} u\|$:

$$c(\mathbf{e}) = \|\langle \vec{\nabla}_1 u(v_0) + \vec{\nabla}_1 u(v_1) \bigotimes \vec{\mathbf{e}} \rangle\| \qquad (4.10)$$

v_0 and v_1 being the two extremities of $\mathbf{e}$. We find the path – between the two borders of the cylinder – which minimizes this functional using Dijkstra's algorithm.

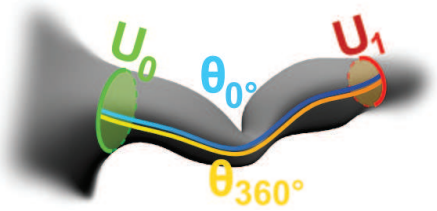


Figure 4.6: Cylinder cut

(u, v) **disk parameterization.** For (topological) disk regions, we directly compute a planar parameterization. Let B be the border of such a given disk. We start by defining a θ-parameterization of B before computing (u, v)-coordinates by solving the linear system:

$$\begin{cases} u(v) = \cos(\theta(v)) & \forall v \in B \\ v(v) = \sin(\theta(v)) & \forall v \in B \\ \triangle u(v) = 0 & \forall v \notin B \\ \triangle v(v) = 0 & \forall v \notin B \end{cases} \qquad (4.11)$$

4.4.2 Bones' geometry

Since one of the main applications we target is skeletal signal processing, we need to make sure that the skeleton's geometry we obtain is not only visually smooth, but has mathematical guarantees about its smoothness. We present our construction below, and discuss it in section 4.4.3.

Starting from the cylinder parameterization, we derive its bone's analytical form $b : [0; 1] \rightarrow \mathbb{R}^3$ as the mean of u-iso-values. We compute it by contouring a given u-iso-value as a polygonal iso-curve made of iso-edges crossing the triangle edges (see Fig. 4.7(a)). We then consider a weighted combination of these iso-edges' centers. Our experiments show that accounting for distortion is the foremost concern. Therefore, the analytical form of b boils down to:

$$b(u) = \frac{\int_0^{2\pi} p(u, \theta) d\theta}{\int_0^{2\pi} d\theta} \qquad (4.12)$$

This simple strategy is insensitive to noise and tends to define the most "natural" bone. See Fig. 4.7 for an illustrative comparison between spatial weighting (Fig. 4.7(b)) and our

integration procedure (Fig. 4.7(c)). Note that the spatial weighting is used in almost every Reeb Graph embedding technique. In practice, we represent bones using interpolating B-Splines with control points sampled from Eq. 4.12.

4.4.3 Bones' regularity

Fig. 4.8 (left) recalls the definition of a 2-manifold and the regularity of a function defined over it: μ is defined over M and locally maps each open set of M to an open set of $\mathbb{R}^2$. These sets recover M in a way that the transition map $\mu_{|U} \circ \mu_{|V}^{-1}$ from $U \cap V \subset \mathbb{R}^2$ to $\mathbb{R}^2$ is regular, for each open set U and V such that $U \cap V \neq \emptyset$. The regularity of M as a 2-manifold is defined as the one of the transition maps. Therefore the function f can be seen through local coordinates given by μ, and $\hat{f}$ is defined by $f = \hat{f} \circ \mu$ everywhere. The regularity of the function f is defined as the one of $\hat{f}$.

In our case, the regularity of the function b depends on the regularity of its related cylinder surface mesh and on the regularity of the (u, θ)-parameterization. In fact, b has the same regularity as the position function on the mesh seen through the (u, θ)-coordinates $\tilde{p} = \hat{p} \circ \hat{\lambda}^{-1}$ since $b(u) \sim \int_{\theta=0}^{2\pi} \tilde{p}(u, \theta) d\theta$ (see Fig. 4.8).

4.4.4 Connecting bones

Once each bone is created, we obtain a *disconnected* skeleton (see Fig. 4.9). Although one can enforce additional constraints on the (u, θ)-parameterization to artificially join their extremities by construction, this usually leads to lower skeleton quality. Instead, we explicitly connect the independently constructed bones by joining the extremities of the ones related to adjacent regions on the triangular mesh.

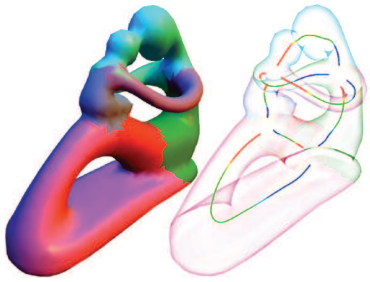


Figure 4.9: Disconnected skeleton

Given such an *articulation*, the adjacency graph $\mathbb{AG}$ of the corresponding cylinders is built. All related bones' extremities are merged, introducing a minor deviation from the original bones: the splines defining them with border U_0 (resp. U_1) are updated with their first (resp. last) control point being relocated at the barycenter of the previous extremities. The induced skeleton deformation can be controlled by setting the domain of the original spline to be unchanged.

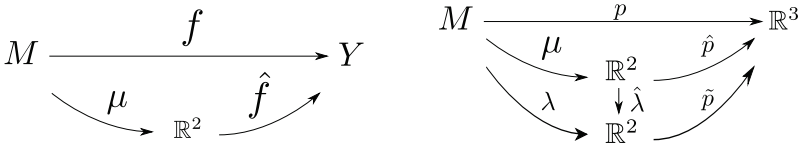


Figure 4.8: **Left:** The regularity of a function f on a manifold is defined as the one of $\hat{f}$. **Right:** The position function p is defined over the 2-manifold M. λ is the (u, θ)-coordinates function defined over M.

4.5 Optimization

In the previous sections, we presented an algorithm to find topological cylinders in shapes and a method to construct a curve skeleton from them. For some applications, it is also important to provide a surface decomposition with well shaped, smooth boundaries We now present the third (optional) component of our approach to iterate over the process, in order to obtain such smooth boundaries.

4.5.1 Surface-restricted skeletal Voronoï diagram

Inspired by the Lloyd relaxation algorithm, we propose to optimize the smoothness of our segmentations in an iterative process. Given the curve skeleton, we compute a new surface decomposition by associating the index of the closest bone of the skeleton to each vertex of the mesh. This corresponds to euclidean space subdivision based on individual bones, therefore defining a *Surface-Restricted Skeletal Voronoï Diagram* (*SRSVD*). This *SRSVD* segmentation is interleaved with our skeleton construction technique in an iterative global relaxation (see Fig. 4.10). On the contrary to the initial step, the segmentation of the following relaxation steps is not necessarly made of topological disks only, but contains topological cylinders as well. Since the *SRSVD* segmentation does not guarantee to preserve all the cylinders generated at previous steps, nor prevent from introducing regions with different topology, we adopt a parametric approach to propose a topology-preserving solution.

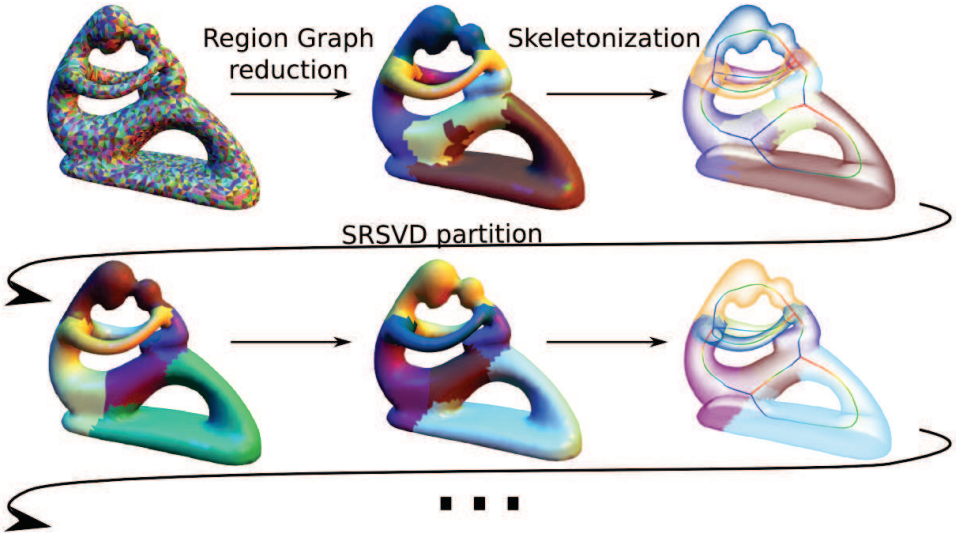


Figure 4.10: Each row corresponds to one step of our relaxation algorithm. The input for the next step is defined using *SRSVD* segmentation.

4.5.2 Parametric skeletal Voronoï diagram

To guarantee a valid segmentation from the *SRSVD*, we parameterize the projection process and consider cylinders/bones at previous step. More precisely, we intersect the Skeletal Voronoï Diagram with the model's surface parameterized by sliding the surface towards its projection onto the previously defined bone: $\forall v : \hat{v}_\alpha = \alpha \cdot b(v) + (1 - \alpha) \cdot v$. By taking $\alpha \in]0, 1[$, the process of smoothing the boundaries is slower, but we have guarantees over the topology of the resulting segmentation. Defining an optimal value for α at each iteration of the algorithm remains an open problem. In practice, we sample $[0, 1]$ uniformly, and take the smallest value that preserves the topology of the resulting segmentation.

4.6 Results and discussion

4.6.1 Performances

Results are presented in Fig. 4.11 for various shapes. The skeletons were obtained with 2 iterations in average. Our curve skeletons faithfully reproduce shape topology and geometry (through regularity). We have implemented the entire algorithm in C++ on an Intel Core2 Duo running at 2.5 GHz with 4 GB of main memory. We report detailed timing for various meshes in Table 4.2.

4.6.2 Properties

Our curve skeleton model satisfies major curve skeleton quality criteria [20]:

Thin our skeleton is analytically defined as a 1-dimensional object

Component-wise this property can be assessed visually on the different illustrations

Efficiency the bottleneck of the algorithm is the region graph reduction, which can be significantly sped-up using prepartitioning (see Table 4.2)

Model		**Timing (in sec.)**		
Name (Triangles)	Patches	Reg. Graph (Edge Coll.)	Skel.	Total
Armadillo (345 944)	9 370	11.0 (66k)	9.9	21.8
Neptune (56 112)	56 112	31.0 (295k)	1.2	32.7
	2 730	2.6 (20k)	1.1	4.0
Memento (52 550)	52 550	30.2 (290k)	1.2	32.9
	3 897	3.6 (29k)	1.1	5.0
Hand (23 464)	23 464	11.7 (132k)	0.4	12.9

Table 4.2: Analytic Curve Skeleton generation time.

Criterion	AS	LS
Analytical form	+	-
Attachement to the surface	+	-
Post smooth editing of the skeleton geometry	+	-
Robustness on non-organic/noisy shapes	-	+

Table 4.3: Our model (AS) compared to [4] (LS).

Nonetheless, we stress the fact that *smoothness* and *regularity reproduction*, although not mentioned in Cornea et al. 's list [20], are fundamental properties for skeletal geometry editing and processing. We refer to section 4.4 regarding good properties of our skeleton in this context.

4.6.3 Comparison

We compare our skeleton model to Laplacian skeletons [4] (results obtained with the authors' implementation, using default values) and Reeb Graphs in Fig. 4.12, as they use the same input, and are commonly used in different applications.

Laplacian skeletons are defined as a contraction of the input mesh, contraction made by successive edge-collapse. As a result, one vertex of the skeleton corresponds to a collection of mesh vertices, usually describing a cylinder as the authors underline in their paper (Fig. 4 in [4]). But this property is not enforced at the core of the algorithm, and we observe that a **lot** of bones in the obtained skeletons do not represent any cylindrical shape of the input geometry (see Fig.4.12, second row). This makes Laplacian skeletons not suitable for the applications we target (e.g., one dimensional model representation for skeletal signal processing, see chapter 5).

Reeb Graphs [68, 74] are defined as the contraction of scalar function iso-contours over a surface. By definition, Reeb Graphs shares with our skeleton the property that one bone of the reconstructed skeleton corresponds to one generalized cylinder in the input geometry. However, Reeb Graphs' geometry depends heavily on the chosen scalar function, which in general does not reproduce a faithful component-wise, surface-related geometry. See Fig. 4.12, third and fourth rows, for examples of skeletons extracted as the Reeb Graph of different functions. The existence of at least one maximum and one minimum of the function implies directly that the extracted skeleton contains at least two extremities, which limits the space of possible structures Reeb Graph can model (see "Fertility"-2^{nd}column in Fig. 4.12 for an example).

4.6.4 Limitations

First, one limitation of our algorithm is its restriction to manifold meshes: extension to point clouds and polygon soups is an interesting direction for future work. Second, our technique can suffer from poorly sampled surfaces with high genus during the region graph *local* reduction. Nevertheless, remeshing or *global* reduction can overcome this issue. Third, *medialness* is not enforced in our algorithm, and some parts of the skeleton may still in-

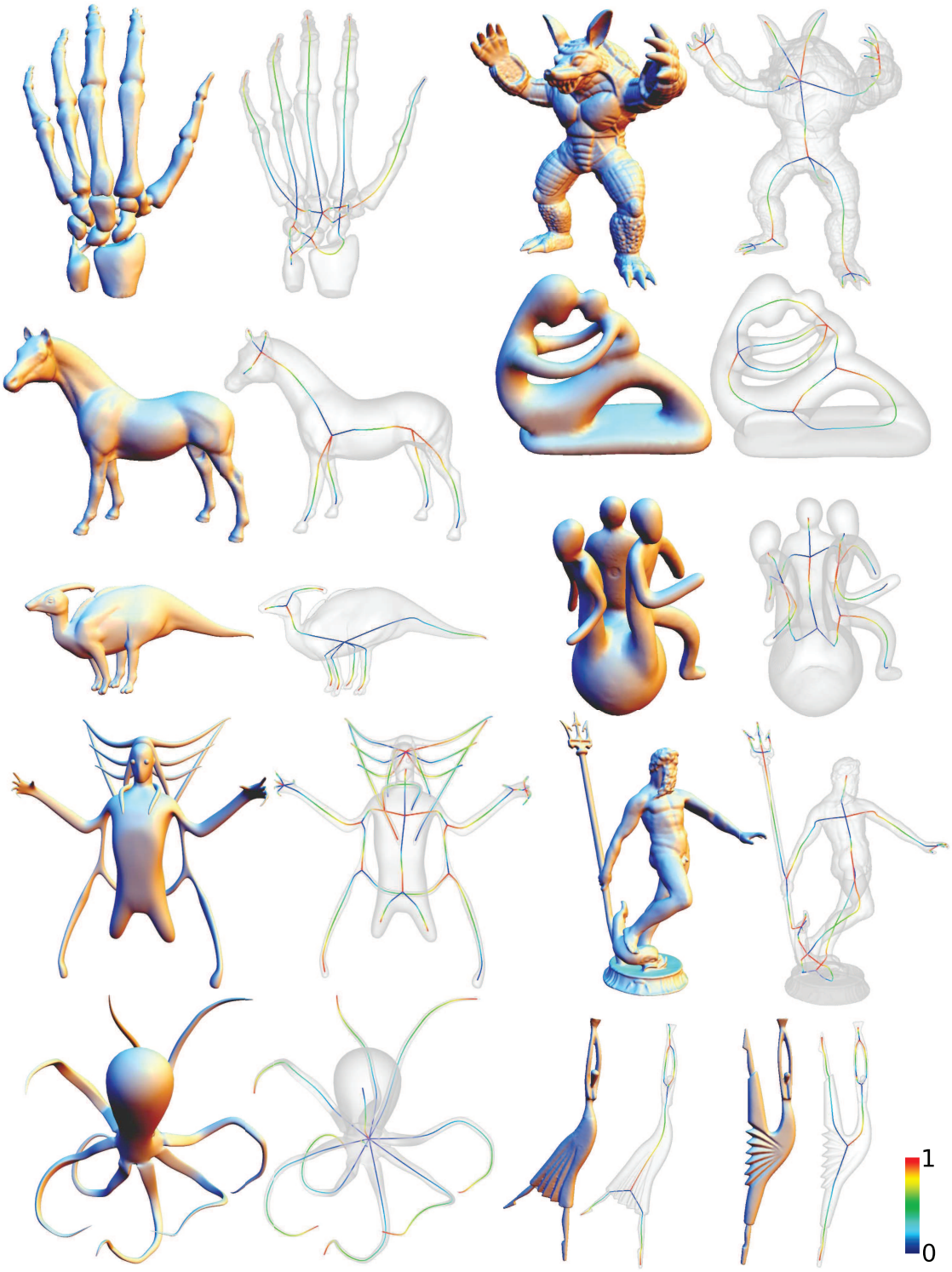


Figure 4.11: Analytic Curve Skeletons of various shapes. The rainbow color code displays the **parameterization** values from 0 (blue) to 1 (red).

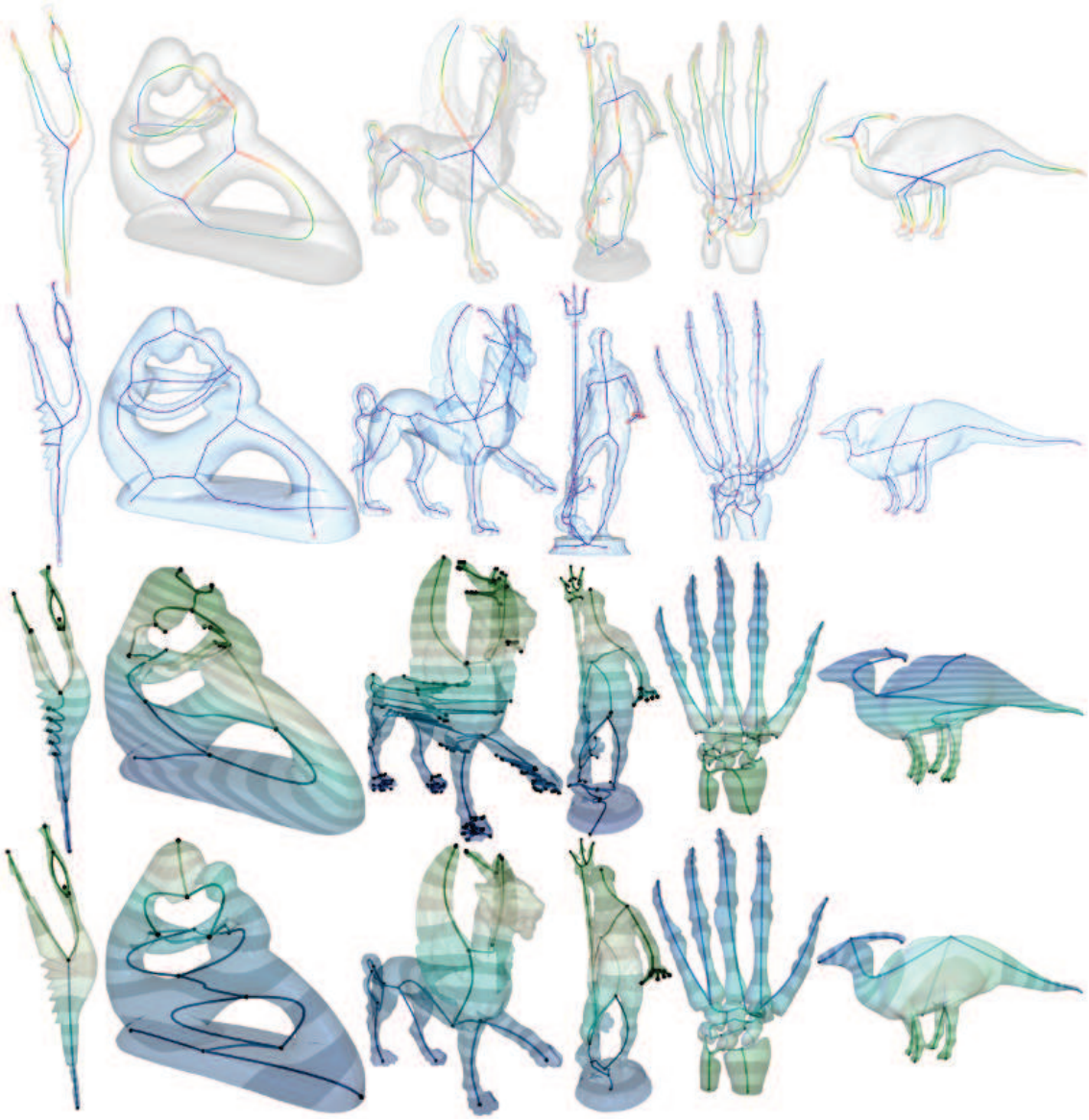


Figure 4.12: **Side-by-side comparison. Top row:** Our skeleton model. **Second row:** Laplacian Skeleton [4]. **Third row:** Reeb Graph on 'Z' function. **Bottom row:** Reeb Graph on harmonic function with user prescribed constraints.

tersect in some places the mesh. This can have an impact on the presented applications. Fourth, although having too many branches is usually not suitable for the typical applications we target (e. g. requiring a *smooth* base domain), it could be interesting to allow the user to refine the skeleton in a multiresolution fashion, to favor particular locations, while keeping coarse the others. Last, as with all curve skeletons, there are shapes which are not suitable for skeletal analysis or processing and which would benefit more from alternative internal structures (e. g. medial surfaces).

4.7 Conclusion

Curve skeletons offer an intuitive and efficient way to track global shape information such as topology, thickness and global alignment of structure on a large variety of shapes. Our Analytic Skeleton model is generated efficiently by means of a disk-cylinder surface decomposition followed by the geometrical integration of per-region harmonic maps. The related surface decomposition can then be improved at boundaries using a specific relaxation method.

In the next chapter, we illustrate the utility of such a parametric skeleton on geometry modeling and processing examples. We believe that a large variety of new applications can benefit from it. In particular, automatic processes such as shape analysis, compression and visualization can make use of its surface-inherited regularity; while animation, interactive shape modeling and processing can exploit its parametric definition and explicit relationship to the surface.

Chapter 5

Skeletal geometry processing

In the previous chapter, we presented a method to obtain an analytic curve skeleton model from an input shape, with a dense correspondance between both entities. We will now present some applications that benefit from our skeleton model. These applications are modeling tools and are essentially based on the smooth *displacement field* that is obtained everywhere by going in the direction of the projection of the vertex on the skeleton. Note, that this set of applications actually drove the formulation of our skeleton model.

Beyond animation, recognition and visualization, geometric modeling and processing have strongly motivated our work. We illustrate the effectiveness of AnaSkel with 3 examples for such applications.

5.1 Skeletal shape modeling

Many applications can benefit from a smooth definition of the "near inside" and "near outside" of a shape, information usually given by the normal of the surface. In Fig. 5.1 we present the Armadillo model and several variations we created simply by defining a displacement function $d_b(u)$ on each bone b and sliding each vertex v of the mesh along its trajectory by the corresponding value $d_{b_v}(u(v))$.

Figure 5.1: Interactive, spatially varying thickness editing using parametric displacement mapping from our skeleton.

The trajectories from the vertices to their projection on the skeleton offer a more consistent way to use displacement maps than the normals of the mesh which, in case of large displacements values, often lead to (local) self intersections of the surface. It also gives a consistent structure for a 0 volume that acts as a "barrier" when applying *negative* displacement (inside the shape). With our solution, the user can also intuitively control the induced deformation "along" components.

5.2 Inset surface modeling

Inset surfaces allow to generate internal structures starting at the boundary surface of a shape and our Analytic Skeleton helps to design them intuitively.

First, the skeleton's geometry and the surface-skeleton trajectories can be edited in a simple way: the user defines a smooth bijective function $I_b(\cdot)$ to change the definition of the u-coordinate onto the bone b. The new coordinate is defined as: $\tilde{u}(v) = I_b(u(v))$ for any vertex v. This $\tilde{u}$-coordinate is not harmonic anymore, but gives the user the ability to edit the trajectories intuitively. For instance, in Fig. 5.2, the user "slides" the trajectories along the bones, to smooth them or to create singularities – in this particular example, trajectories starting from the head of the mother all point to the same location after editing.

Second, the user can edit smooth inset surfaces with local control of its shape, while preserving the natural layout of the vertices on the bones, this layout being induced by the initial harmonic u-parameterization. The inset surface is then defined as a linear interpolation between the input surface and its skeleton, with $\alpha \in [0;1]$ an interpolation parameter that can vary spatially (i. e. for each vertex):

$$IS_\alpha(v) = (1-\alpha)\cdot \mathbf{p}_v + \alpha \cdot b(\tilde{u}(v)) \tag{5.1}$$

Examples of interactively edited inset surfaces are shown in red in Fig. 5.2 (top).

Due to the parametric nature of our analytic skeleton, painting offers a simple means to edit the geometry of the skeleton as well as the shape of inset surfaces. The user draws using a brush equipped with smooth functions $W_i : M \Rightarrow \mathbb{R}^i$ that act as a displacement modulation from the surface to the skeleton or vice versa. This function is inserted in Eq. **??** to obtain $b(u) = \frac{\int_0^{2\pi} W_1(v(u,\theta))\cdot p(u,\theta) d\theta}{\int_0^{2\pi} W_1(v(u,\theta)) d\theta}$, affecting smoothly the skeleton's geometry.

User-supervised weighting can also be integrated in Eq. 5.1 to obtain $IS_t(v) = (1 - Spl(v)(\alpha)) \cdot \mathbf{p}_v + Spl(v)(t) \cdot b(\tilde{u}(v))$ and locally deform the inset surface by editing its local depth. The weighting function is then in the form $Spl : M \Rightarrow \{F_n : [0;1] \rightarrow [0;1]\}$, F_n being a bijective cubic spline of order n.

As an application, we propose an interactive Shell Mapping [71] tool, with which the user can "peal" surfaces using one of its specific inset surfaces and then map arbitrary geometry to the so-defined in-between shell space (see Fig. 5.2 bottom).

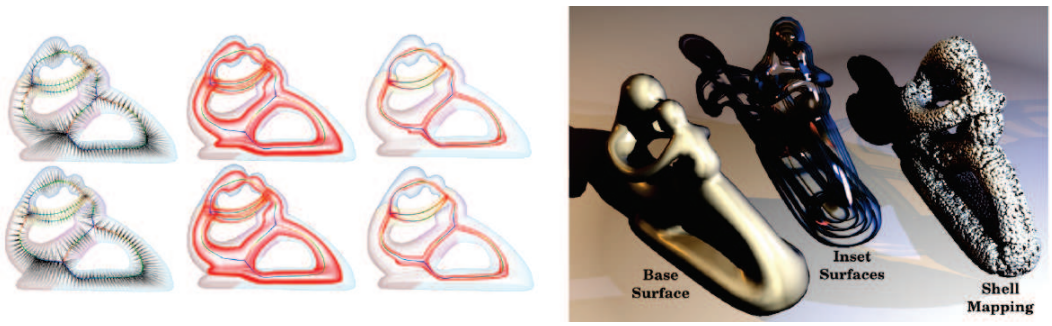


Figure 5.2: **Top:** Smooth inset surfaces extraction and interactive editing. **Bottom:** Interactive design of in-between shell space for mapping of arbitrary, even procedural geometry (right) without inflating the global volume of the shape.

5.3 Skeletal mesh filtering

Mesh smoothing is a fundamental tool in geometry processing. Beyond low-pass (Laplacian) filtering, the Bilateral Filter has recently emerged as an efficient feature-preserving operator. Originally designed for images [93], it has been adapted to meshes and higher dimensional data. Here, the basic idea is to replace the standard spatially weighted local combination of samples by bilateral weights accounting for both sample proximity in the domain and in the range of the sampled function. Although domain and range spaces are clearly defined for images (i. e. , pixel positions and pixel colors), this notion is ill-posed for geometry, as 3D vertex positions are usually not expressed relatively to a global parametric domain. Jones et al. [45] overcome this problem by deducing a range value from the distance to neighbors tangent planes:

$$BL(\mathbf{p}_v) = \frac{\sum_{\mathbf{u}\in N(v)} \Pi_u(\mathbf{p}_v)\omega_s(||\mathbf{p}_u - \mathbf{p}_v||)\omega_r(||\Pi_u(\mathbf{p}_v) - \mathbf{p}_v||)a_u}{\sum_{\mathbf{u}\in N(v)} \omega_s(||\mathbf{p}_u - \mathbf{p}_v||)\omega_r(||\Pi_u(\mathbf{p}_v) - \mathbf{p}_v||)a_u}$$

with a_u an area weight, ω a Gaussian kernel or one of its polynomial approximations and the orthogonal projection $\Pi_u(\mathbf{p}_x) = \mathbf{p}_x - \langle \mathbf{p}_x - \mathbf{p}_u | \mathbf{n}_u \rangle \mathbf{n}_u$ modeling *coplanar similarity* as a range space.

Although this definition works perfectly well for small scale high frequency features, it quickly loses its detail preserving behavior for mid-size structures. Basically, the coplanar similarity assumption does not hold for larger structures and we propose to replace it by a local *thickness* similarity measure. To do so, we make use of our curve skeleton to define a new filtering operator, more suitable for larger structure preservation: *Skeletal* (Bilateral) Filtering.

Let $\Psi(v) = b(u(v))$ be the projection of $\mathbf{p}_v$ onto its related bone. Basically, we model the shape thickness similarity r as the norm of the surface-skeleton vector $\Psi(v) - \mathbf{p}_v$, establishing the skeleton as the domain and Ψ^{-1} as the geometric signal (range). We can therefore express skeletal filtering as:

$$SBL(\mathbf{p}_v) = \frac{\sum_{\mathbf{u}\in N(v)} \Pi_u(\mathbf{p}_v)\omega_s(||\mathbf{p}_u - \mathbf{p}_v||)\omega_r(|r(\mathbf{u}) - r(v)|)a_u}{\sum_{\mathbf{u}\in N(v)} \omega_s(||\mathbf{p}_u - \mathbf{p}_v||)\omega_r(|r(\mathbf{u}) - r(v)|)a_u}$$

Fig. 5.3 illustrates the differences to standard bilateral filtering: medium size structures are preserved even if small structures are strongly smoothed out. This behavior is often

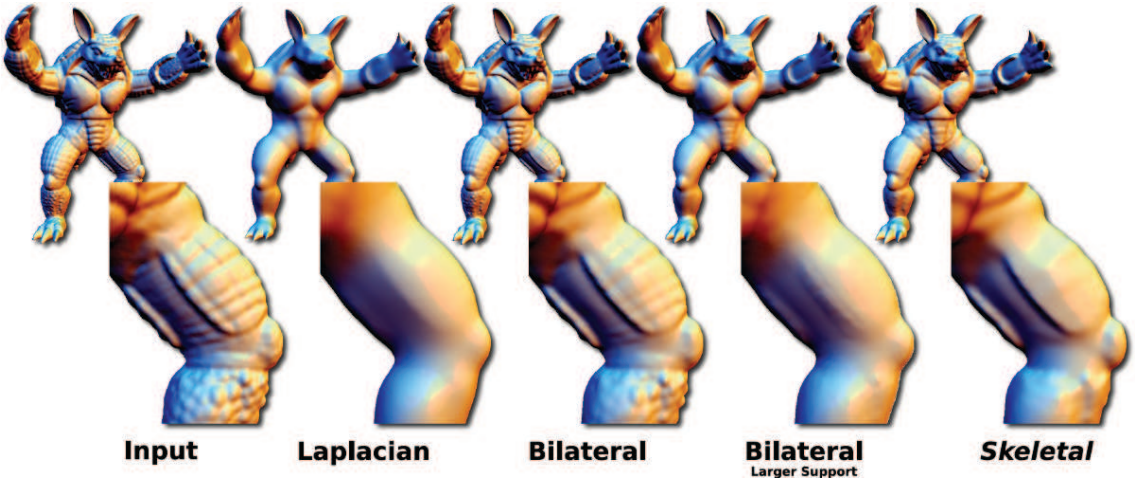


Figure 5.3: Surface filtering comparison. All pictures but the fourth use identical support sizes.

desirable when processing shapes with a wide range of frequencies. The skeletal filter can be used in conjunction with laplacian and bilateral filters in a multi-lateral filter, where planarity and thickness act as complementary range spaces.

5.4 Discussion

By nature, curve-skeletons are not differentiable at the joint points where more than two bones connect (it is not even locally 1-manifold). This property has to be taken into account when designing skeleton-based applications. The skeletal mesh filtering application (see section 5.3) is not affected by this, as demonstrated in the results where the filtering has been performed *globally* on the mesh, and not only onto a bone. The inset surface modelling application (see section 5.2) is not affected either, as the result of this application is used to construct a shell-space embedding, which does not require necessarily smooth boundaries everywhere. On the contrary, the displacement application (see section 5.1) requires to set a displacement that is *regular* in order to avoid strong normal discontinuities at the boundaries of the cylinders. This is achieved in our framework by adjusting the displacement function derivatives at the joints of the bones.

Manual editing of the AnaSkel Up to this point, we presented an automatic method to compute the AnaSkel of a triangular mesh. We eventually came to think that: (i) the simplicity of the curve skeletons' representation implies that they shoul be editable easily by an artist, (ii) the optimal topology and geometry of a surface skeleton are very dependant to the artist's particular wishes, and (iii) the optimal topology and geometry of a surface skeleton are very dependant to the application targetted by the artist.

All three points together motivate the possibility of editing manually a surface skeleton. Our framework could allow easy interaction with the creation process of the AnaSkel, since it is built upon a cylinders & disks decomposition, and brushes allowing to design cylinders have been developed already [104]. Note, that an automatic detection of missing bones corresponding to protuberances on cylinders is already feasable and implemented (see Fig. 5.4).

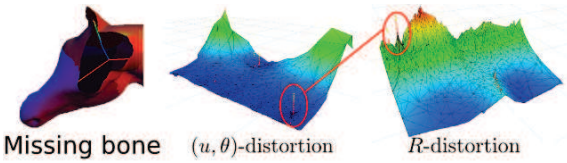


Figure 5.4: Cross analysis of (u, θ) distortion and R (distance to bone) distortion maps on the cylinder over a scale space (right) reveals the missing bone corresponding to the left ear of the horse (left).

Local AnaSkel Many recent modeling softwares, as Blender [1] and especially ZBrush [2] are *brush-based*. In particular, during a standard modeling session, a smooth base mesh is very often used, and geometric detail is added on top of several hierarchy levels by the use of a *displacement brush*. During this process, the mesh can be updated at different levels of the hierarchy. Displacement is performed in the direction of the normal of the mesh, and artists must take care of *self intersections* that are introduced by this operation.

We believe that *topological brushes*, allowing to design interactively smooth decomposition domains of various types (disks, cylinders,...) can help defining smooth directions for displacement mapping. For a cylindrical local domain, the embedding strategy we presented in the previous chapter can be used. The main challenge resides in the automatic definition of this local domain, that needs to be adapted to the geometry and to the scale of the brush used by the artist.

Part III

On-Surface structures

In part II, we presented results and shape modeling applications on structures that were 1-dimensional and *inside* the shape. In this part, we present two kinds of structures that are embeded on the surface, and that allow easy and efficient definition of handles for variational mesh deformation techniques.

In chapter 6, we present our simplicial complex for deformation, which is a multi-resolution structure allowing the user to select either regions, curves or points on the surface.

In chapter 7, we present the construction of curve sets, that are derived from intrinsic and view-dependent properties of the surface, for which we show that they are good candidates for the selection of features on the surface.

Chapter 6

DEX: On-surface simplicial complex for deformation

Linear variational mesh deformation techniques such as LRI (see section 3.3.1 in the technical background) allow to perform deformations on surfaces efficiently. We remind the reader of the modeling metaphor that is used for this technique. Each time the user wants to perform a deformation, a region of influence (ROI) is defined; it is composed of a *handle* that the user grabs and deforms, a *fixed region* that remains still, and an in-between region where the geometry is optimized by the system to minimize some *bending* and *stretching* energies, therefore preserving geometry as much as possible in the deformation process.

While techniques such as LRI have been intensively developed over the past years, the definition of the ROI itself remains a tedious task for the artist and a time consuming process in a typical modeling session.

In this chapter, we tackle this problem and propose the definition of a *deformation complex* embedded on the surface, allowing for a user-friendly selection of handles on the mesh. We rely on the LRI technique for the deformation, but any other variational technique requiring the definition of a ROI for the deformation can benefit of our approach.

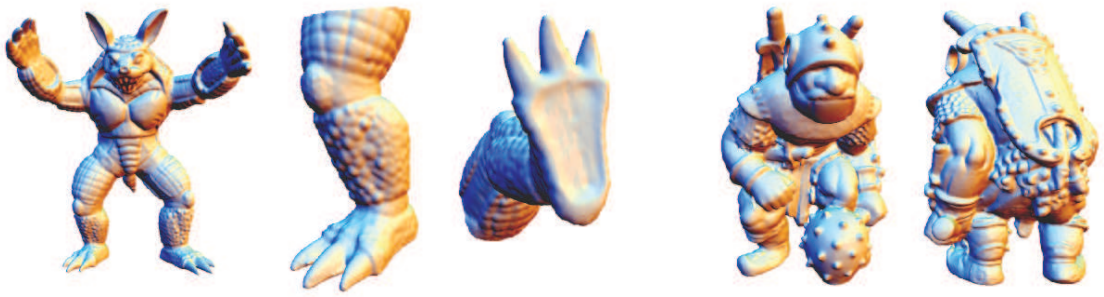


Figure 6.1: Many 3D objects present features at different scales.

6.1 Deformation complex

We propose to define a simplicial complex on the surface as the underlying domain for handle selection. The user can choose to select either a region, a curve separating two regions, or a point joining several curves.

Many 3D objects present features at different scales (see Fig. 6.1). We adopt a strategy where the user can easily switch from one level of detail to another in the deformation complex (e. g. using the mouse wheel) in order to adjust the scale of the feature he or she grabs. The deformation complex is therefore built in a multi-resolution fashion.

Regions that the user grabs should be homogeneous in some way, and it should correspond to a feature. Based on this observation, we cast the problem of the definition of the deformation complex to the definition of a multi-resolution segmentation of the surface. The complex is built on top of this segmentation. The adjacency information in the regions is used to define the free part of the ROI from the only information of the handle.

Our deformation pipeline is illustrated in Fig. 6.2, and is composed on two main steps: (i) the definition of a multi-resolution segmentation, based on a set of multi-resolution descriptors on the mesh, and (ii) an interactive step, where the user can navigate over the different levels of the hierarchy and use the resulting deformation complex to edit the shape.

6.2 Multi-resolution segmentation

This multi-resolution segmentation is performed on different energies at different scales, with an increasing number of clusters when increasing the level of detail.

Using the k-means algorithm, we cluster the mesh triangles into k_l regions at level $l \in [1, L]$ ($k_l < k_{l+1}$), minimizing the standard deviation of different descriptors. The size of the regions should decrease when increasing the level of detail, and therefore the number of regions should increase at the same time.

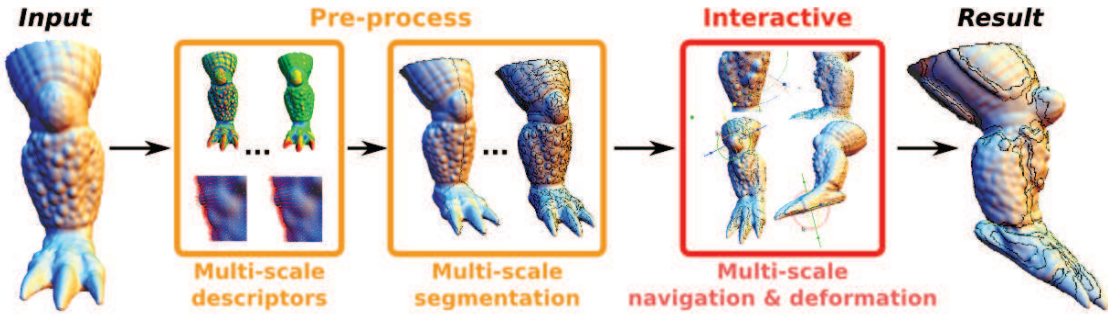


Figure 6.2: DEX pipeline

6.2.1 Variational shape approximation

We quickly review the variational shape approximation technique from Cohen-Steiner et al. [19], that is an adaptation of the k-means algorithm on surfaces, that uses a relaxation similar to Lloyd's strategy [65].

The input is a manifold triangle mesh $\mathbf{M} = \{\mathcal{V}, \mathcal{T}\}$ with appropriate topology, an integer k being the number of clusters in the partition, and a **metric** $\mathbf{d}(\cdot, \cdot)$ that defines the distance from a triangle to a cluster.

The different clusters are equipped with so-called **proxy** geometry, which is a simple plane defined by a point p_i and a normal n_i. The algorithm is iterative, and works as follows:

- **[Initialization]:** Pick k triangles in the mesh as *centroids*, and define the geometry of the corresponding proxies using the centroid of the triangles and their normal;
- **[Step 1]:** Using a **priority queue**, floodfill the regions on the mesh using a breadth-first strategy – with a cost defined by the metric $\mathbf{d}(\cdot, \cdot)$: a triangle t_i is suggested to be added to the region r_j with **cost** $\mathbf{d}(t_i, r_j)$;
- **[Step 2]:** Update the geometry of the proxies as the plane that fits best the geometry of the clusters;
- **[Step 3]:** For each cluster, set its triangle that deviates the least from the updated proxy to be the new centroid;
- If convergence is not reached, go to step 1.

Two different metrics are presented in the original article, the $\mathcal{L}^2$ metric that gives the squared distance to the tangent plane of the proxy, and the $\mathcal{L}^{2,1}$ metric that gives the squared difference between the normal of the triangle and the normal of the proxy.

$$\begin{cases} \mathcal{L}^2(t_i, r_j) = Area(t_j)||t_i - \Pi_j(t_i)||^2 \\ \mathcal{L}^{2,1}(t_i, r_j) = Area(t_j)||n(t_i) - n(r_j)||^2 \end{cases} \tag{6.1}$$

The authors show several advantages of using the $\mathcal{L}^{2,1}$ metric over the $\mathcal{L}^2$ metric (see Fig. 6.3). Convergence is reached faster, and errors are diminished (when comparing the original mesh and remeshed version based on the proxies that are found). Of course, as the normals of the triangles are very easily disturbed by noise from the position of the mesh vertices, **normal smoothing** may be necessary in some cases to obtain visually pleasing results.

Complexity The segmentation algorithm that we use has a complexity that is dominated by the complexity of the flooding procedure, which is $N_T \log(N_T)$ (N_T denoting the number of triangles in the mesh). It is relatively insensible to the number of clusters k_l, and runs at interactive rates on meshes that we presented in Fig. 6.1. This concerns only the pre-processing step, and the interactive exploration of the deformation complex that we introduce later in section 6.3 is performed in real-time.

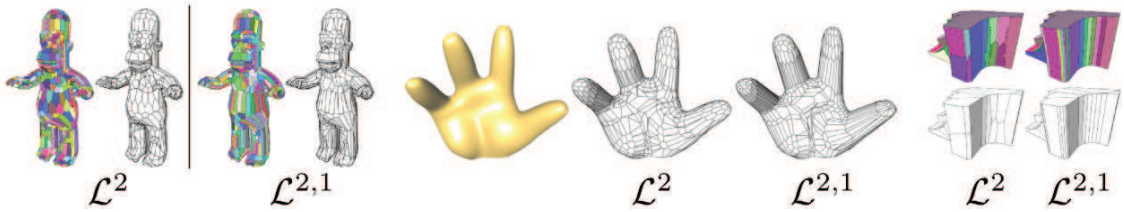


Figure 6.3: Comparison of the $\mathcal{L}^2$ and $\mathcal{L}^{2,1}$ metrics for the variational shape approximation. *Image taken from [19].*

6.2.2 Multi-resolution descriptors

Our clustering of the mesh is based on a set of descriptors on the triangular surface. Following the idea of Cohen-Steiner et al. [19], we use *the normal of the triangles* as a first descriptor to perform the clustering at coarse levels.

Our experiments show, that using the curvature as a descriptor for fine levels of details leads to segmentations that are *complementary* to the low-resolution segmentations.

To summarize, our descriptors composed of two terms:

- a normal per triangle (used generally at coarse levels)
- a curvature value per triangle (used generally at fine levels)

We first compute the descriptors per-vertex. We then set the descriptors of each triangle as the average of the descriptors of its three vertices. Even if the normals of the triangles are perfectly defined (in contrast to per-vertex normals), this strategy tends to give better results, as it smoothes the descriptors naturally, resulting in a partitioning that is less sensitive to noise. As we perform our partitioning in a multi-resolution fashion, the definition of the descriptors at the first scale (no blending) are compatible with the ones at coarser scales.

To obtain our multi-resolution segmentation, we compute a k_l-means clustering at each level l, that minimizes the deviation to the chosen-per-level descriptor, using the algorithm presented in section 6.2.1. We finish our partitioning by applying a regularization scheme on the boundaries of the segmentation, in order to obtain smoother regions. For this, we iterate over the triangles of the mesh that are on the boundaries of the different regions, and we attribute the triangle to its neigbooring region if two of its edges belong to the boundary. We perform this change only if the topology of the resulting segmentation stays the same.

Scale space To retain large features based on normals, features at coarse levels should be based on low-frequency geometry. In order to obtain this property, we compute a *scale space* over the set of descriptors: at each level $l \in [1, L]$, we compute the average of the descriptors of the vertices using a gaussian kernel of standard deviation $\sigma_l = \frac{(L+1-l)\sigma_{max}}{L}$ over a sphere of radius $3\sigma_l$ (this value is commonly accepted as a good fixed support for

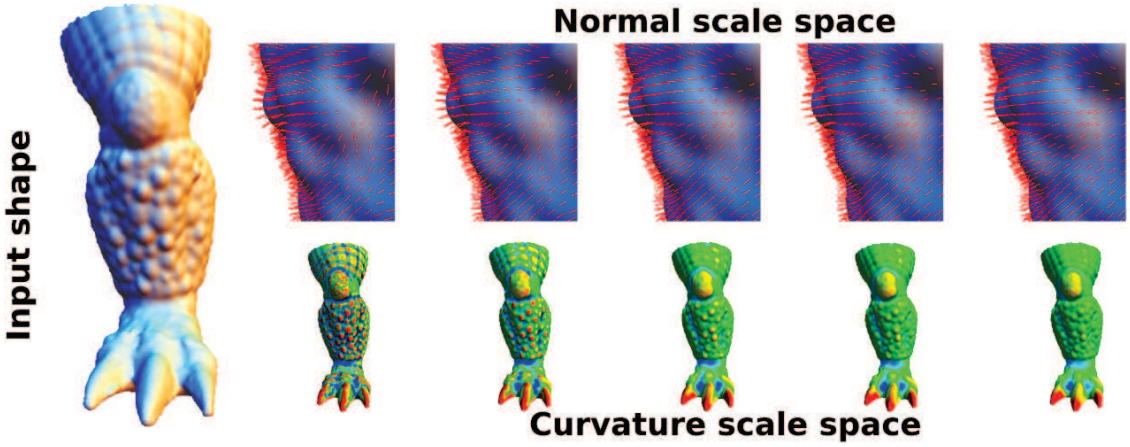


Figure 6.4: Descriptors of the vertices are blend over the scale space.

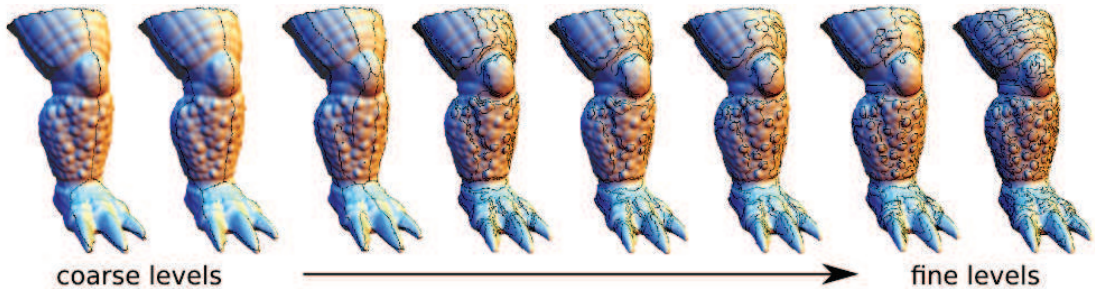


Figure 6.5: Multi-resolution segmentation.

gaussian kernels, as it retains more than 97% of the information). The normals in the descriptors are renormalized after blending.

The gathering of the neighborhood is performed by flood-filling on the mesh starting at an initial vertex, to ensure that we blend our descriptors over a connected component of the mesh. For small enough neighborhoods, at high levels of the scale space, the obtained set is equivalent to a disk. To some extent, the distances are equivalent to geodesics as the neighborhood is geometrically close to the tangent plane of the vertex. For large neighborhoods, at low levels of the scale space, these two properties don't hold anymore.

Note, that the computation of the scale space is performed as a pre-process. An example of the resulting segmentation is presented in Fig. 6.5.

6.3 Interface for deformation

Based on our multi-scale segmentation, we construct the deformation complex at each scale by considering the adjacency of the regions on the triangle mesh. A simple strategy that we adopt is the following:

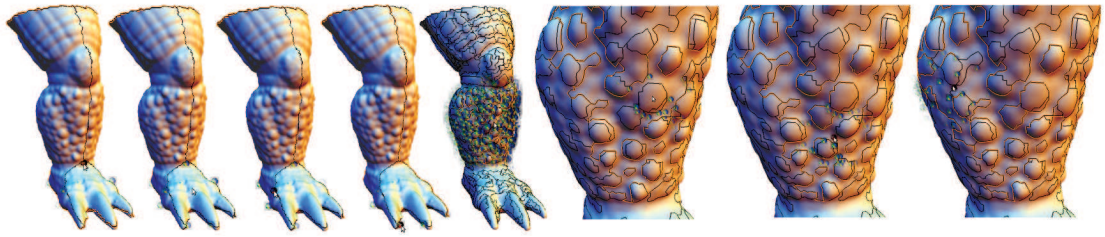


Figure 6.6: **Exploration of the deformation complex:** The user can grab *regions*, *curves* or *points*. The region of interest (ROI) of the deformation is adapted to the selection. Selectable features are displayed w.r.t. the zoom in the scene, and is reduced to a certain area around the mouse for visualization.

1. the handle part of the ROI is composed only of the vertices that belong to the selected feature (either a region, or a curve, or a single vertex);

2. the vertices that are located on the boundary curves of the union of the regions adjacent to the selected feature compose the fixed part of the ROI;

3. the vertices in-between compose the remaining part of the ROI, and only their position is unknown when setting up a linear system for the deformation.

We show in Fig. 6.6 an example of exploration of the deformation complex. Small images on the mesh show the curve and point features that can be selected and are "mouse grabbers" that can be clicked on by the user to select and deform them. For the curves, the "mouse grabber" is located at the middle of the curve (except for cycles, where it is put arbitrarily). To select a region, the user can click directly on it.

In a typical modeling session, the user can edit features at different scales, independently of the order in which he chooses the scales. In Fig. 6.7, a standard modeling session using our deformation complex is illustrated: in this case, *fine scale* features (bumps on the leg), then *coarse scale* features (foot) and finally *mid-scale* features (knee) were edited successively. It took less than thirty seconds to execute this standard modeling session (after DEX definition).

6.4 Discussion

First, as we build our deformation complex over a segmentation, we make the assumption that every part of the mesh is of some interest for the selection of features. This is not true in general, but our formulation is useful for a wide class of meshes that contain features of very different scales.

Second, the energies that we minimize are limited to representing a certain class of features (large quasi-planar parts at low scales and bumps and local extrusions at higher scales). The segmentation process that we use is also based on a local gathering, and a more global strategy could allow better results.

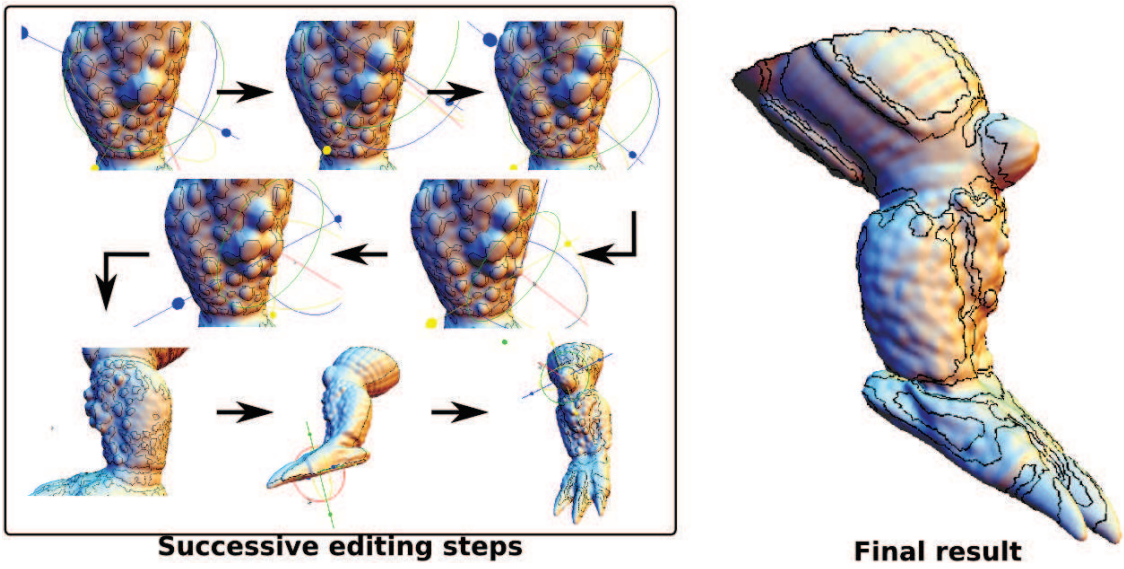


Figure 6.7: **Manipulation of the deformation complex:** The user edits features of the mesh that were selected by our multi-resolution approach. The order in which he or she selects the scale of the deformed feature is not important.
The final result of the modeling session is displayed on the right.

In the following chapter, we study a complementary approach and show how *curves* and curve-like features defined on the surface can help defining handles for mesh deformation.

Chapter 7

On-surface curve set for deformation

In the previous chapter, we proposed to use the definition of regions as handles for mesh deformation. We note that sometimes, not every part of the shape is meant to be seen as a feature. This motivates the definition of handles on the surface, that do not necessarly derive from a segmentation of the shape.

In this chapter, we present an interface for the selection of *feature curves* as handles for mesh deformation. We propose to derive our feature curves from intrinsic, as well as from view-dependent curvature information using existing line drawing algorithms, as these two classes of curves describe both interesting features to grab, and they are orthogonal.

As for the previous chapter, we rely on the LRI technique for the deformation, but any other variational technique requiring the definition of a ROI for the deformation can benefit of our approach.

7.1 Selection of intrinsic feature curves for the deformation

As seen in the previous chapter, large features can be retained by performing a geometric analysis of the low frequencies part of the surface. In this section, we propose to use the curvature information in a similar way.

7.1.1 Construction and regularization

Fig. 7.1 illustrates the main steps of our pipeline:

1. the curvature is computed on the mesh
2. a subset of the mesh is considered as features (all vertices v such that $\kappa(v) \in [\kappa_1, \kappa_2[$)
3. insignificant features are removed, based on their areas

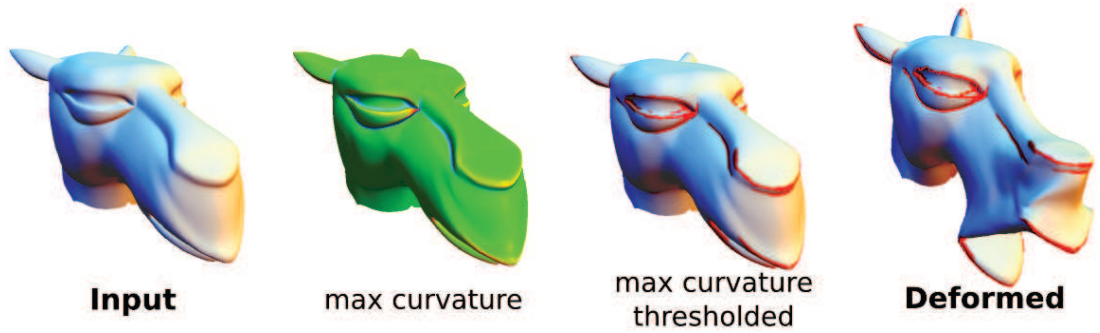


Figure 7.1: We rely on the curvature to detect important features on the mesh, allowing for easy manipulation of the input mesh. Some insignificant features were not taken into account, based on the surface of the feature.

Based on the kind of features the user wants to retain, mean curvature $\mathbf{H}$, maximum curvature $|\kappa_1|$ or the difference between maximum and minimum curvatures $|\kappa_1| - |\kappa_2|$ can be used as the descriptors of the shape's features.

In the example of Fig. 7.1, the absolute value of the maximum curvature $|\kappa_1|$ was used as a descriptor for our features. The red parts on the surface (Fig. 7.1, 3^{rd}) indicate a level set $V_{|\kappa_1|}(s) = \{v \in \mathbf{M} || \kappa_1 |(v) > s\}$ parameterized by a thresold s tuned by the artist. This parameter describes the scale of the feature he or she wants to grab. Some features that were small were ignored in this example, based on their area (we eliminated all features that had an area smaller than 30% of the largest feature).

Note, that smoothing of the curvature on the mesh, as described in the previous chapter, regularizes their level sets, therefore the handles given by our technique. Alternatively, the curvature could be computed at large scales, using quadric-fitting techniques on large neighborhoods. These methods are typically not used for applications requiring precise local curvature, on the contrary to our case where we are interested in retaining only large scale features of the surface.

7.1.2 Interface for deformation

The strategy that we propose for the deformation is straigtforward: when the user grabs a feature, we set the others to be the fixed part of the ROI, and all others vertices in between both parts are optimized using LRI. This strategy involves generally larger systems than when using DEX (see chapter 6).

Note, that we chose to set high values for the thresholding of the descriptors because we are interested in finding large scale feature lines in the mesh (see Fig. 7.1). Large features that are disconnected, yet close on the mesh, can be a problem when being manipulated one by one: the area in-between is not large enough, and by manipulating only one feature and setting the other one to be fixed, the user can create *cracks* on the geometry. Several approaches could be used to solve this problem: e. g. thresholding by hysteresis, non-selection of features that are geodesically close to the handle, setting of different orders of preservation on these features.

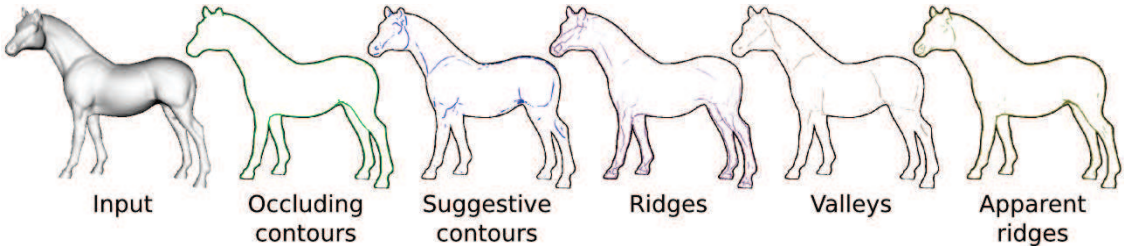


Figure 7.2: Different line drawing algorithms allow to retain details of visual importance on 3D surfaces. These describe geometry that is sometimes difficult to capture from other view points. *Images generated using the RTSC software, distributed by Princeton University.*

7.2 Selection of view-dependent feature curves for the deformation

As written before, we can also use view-dependent feature lines stemming from Line drawing algorithms, such as contours and apparent ridges (see Fig. 7.2). These curves are orthogonal to the class of intrinsic feature curves on the mesh (that are view-independent).

Line drawing algorithms focus on retaining information of the mesh that is pertinent to the users for the visualization of 3D objects. As these algorithms aim at describing features of visual importance, that are very difficult to capture from other points of view, these curves constitute good candidates as handles for mesh deformation.

7.2.1 Line drawing rendering

Line drawing algorithms aim at defining curves that exhibit features of visual importance on a 3D surface (see Fig. 7.2). These can be view-dependent (occluding contours, suggestive contours) or not (ridges, valleys). They are often described as being part of the non-photorealistic (NPR) rendering techniques, as they allow to abstract the representation of the shape on the screen.

These curves are computed from curvature information on the mesh:

- Occluding contours are the set of points on the mesh, whose normal is orthogonal to the view direction.
- Suggestive contours are the set of points that likely to become contours, for small variations of the point of view. Equivalently, they are curves with null radial curvature.
- Ridges (resp. valleys) correspond to curve where the principal curvature assumes a maximum in the pricipal direction, for which $\kappa_1 > 0$ (reps. $\kappa_1 < 0$). Their computation involves the curvature as well as the derivative of the curvature.
- Apparent ridges are the same, but with respect to view-dependent curvature [48].

7.2.2 Curves regularization

Unfortunately, many view-dependent curves are not smooth on the 3D surface, even though they appear smooth to the user once projected on the screen (see Fig. 7.3). Moreover, these curves are sometimes disconnected in a imperceptible way, and composed of small components that are very close to each others. This leads to unexpected deformation behaviour when considering these curves for deformation, as it can introduce cracks in the geometry. We solve these problems by first processing the set of curves using the following strategy:

1. we aggregate small components that are geodesically close on the mesh, on a per-type basis;

2. we remove components that are small on the screen, according to a user-threshold;

3. we smooth the rest of the curves one by one, by iteratively applying a weighted least-squares filtering on them and projecting them back on the surface (using an interpolation MLS operator).

7.2.3 Interface for deformation

We propose to use the resulting set of smooth curves in a different way than before, when we were considering intrinsic features that were large and sparsely distributed on the surface. Here, a more classical approach is used: the user selects a curve as handle by clicking on it; the ROI of the deformation will then simply be the area on the mesh that is geodesically close to the curve. Two parameters can be tuned by the user: (i) the length of the handle (the distance from the selection point) and (ii) the size of the ROI can be adjusted (see Fig. 7.4). When changing these parameters, the visualization of the ROI can be updated in real-time, as their computation requires only flood-fill algorithms on the mesh, in regions that are generally relatively small.

Results of deformations are presented in Fig. 7.5. These examples present some features that are traditionnally tedious to select, and that were straigtforward to capture using our technique (e. g. the contour lines in the double torus object).

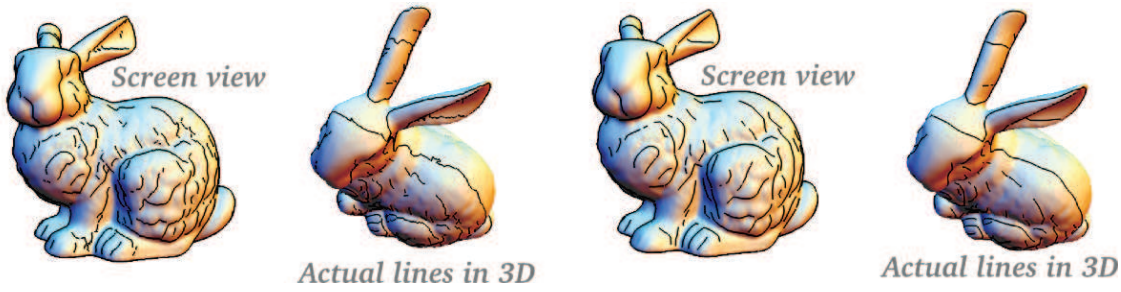


Figure 7.3: Regularization is performed per-curve, and allows to obtain smoother handles for the deformation.

7.3 Discussion

We have presented in this chapter two complementary approaches for the selection of curve handles on meshes, stemming from intrinsic and view-dependent properties of the mesh. In some cases, it is completely natural to sketch deformations from contour curves, as demonstrated in previous work. To cope with the irregularity of the rendered curves, that is invisible to the user, we proposed a regularization strategy that is efficient and allows to obtain the expected deformations.

More advanced deformations could be obtained by taking into account the nature of the different curves: (i) intrinsic, (ii) contour curves, (iii) suggestive contours, (iv) apparent ridges, that correspond to various differential properties of the mesh. In particular, the selection of some curves could be used in the deformation process, not as handles, but rather as specific geometry-preserving constraints. Our approach could also be improved by using advanced hysteresis methods.

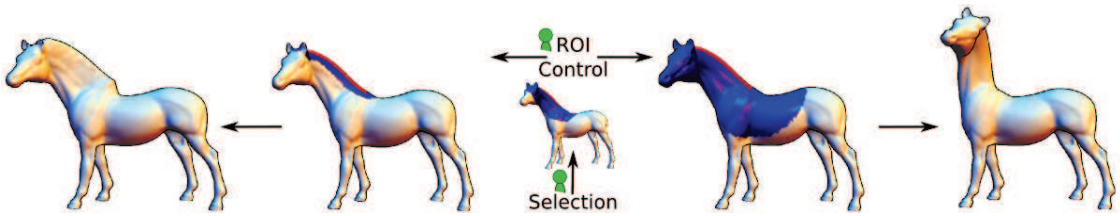


Figure 7.4: Tuning of the influence area of the ROI allows to obtain different effects using the handle for deformation.

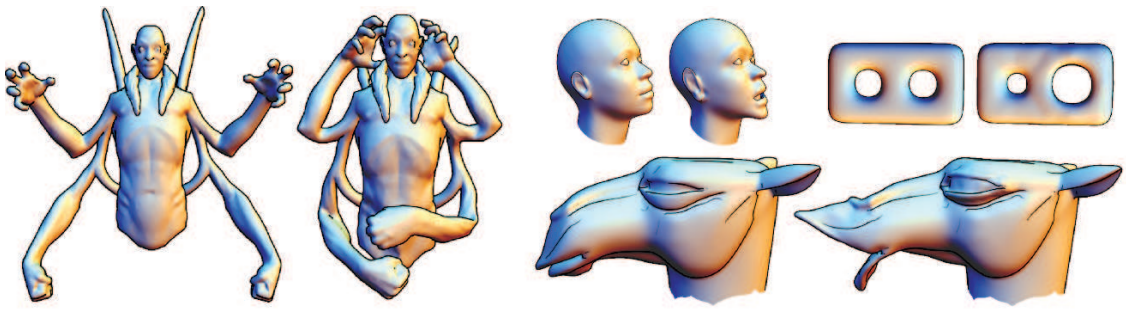


Figure 7.5: Results of deformations using view-dependent NPR curves.

Part IV

Outer structures

In part II and part III, we have presented structures that were 1-dimensional and 2-dimensional, and that were *inside* the surface or *on* the surface. It this part, we are presenting results that benefit from 3-dimensional outer structures, the so-called *cages* that allow the use of a variety of deformation techniques.

In chapter 8, we see how to represent animated 3D shapes stemming from performance capture using cages. The inverse problem is not straightforward to solve, as it presents high condition numbers in general, and requires adapted strategies that we introduce in this work. In particular, we introduce new *relaxation* and *spectral regularization* strategies, and show how these two orthogonal approaches combine.

In chapter 9, we see the possible applications of such a representation, and discuss the advantages and drawbacks compared to previous approaches.

Chapter 8

Cage-based representations of animated shapes

Cage coordinate systems have been studied intensively over the past years. They allow to encode each point of the inner space of a cage w.r.t. its vertices. The main goal in the development of these coordinates has always been to obtain smoother and more beautiful deformations when manipulating the vertices of the cage. The compromise between their simplicity of use and their expressive power has made them a very popular tool in the animation industry.

These sets of coordinates present a high condition number for the inverse problem, and the latter is particularly challenging. Indeed, they were designed to restrain the space of possible deformations of the inner space of the cage, even when manipulating poorly its vertices (for example, Green Coordinates guarantee to obtain quasi-conformal deformations, regardless of the deformation of the cage).

In this chapter, we show how to represent animated 3D shapes obtained from performance capture using cages. We present an intuitive framework for this, that is built upon new *relaxation* and *spectral regularization* strategies. We provide a thorough mathematical analysis of these two orthogonal approaches, and show their complementarity in this particular context.

8.1 Introduction

Motion capture and representation has always been a major concern in the cinema industry. Many recent movies feature virtual characters that need be animated in a realistic way, or even whole virtual worlds where no real-life object is displayed at all. One amusing fact is, that motion capture actually preceded movies, and not the opposite. It is accepted that Eadweard Muybridge was the first to capture images from the real world in order to reconstruct what we later called a "movie". His invention was motivated by a bet: he wanted to prove that a galloping horse had, at some point in the stride, all four hooves off the ground (see Fig. 8.1).

Figure 8.1: The first movie ever shot: Eadweard Muybridge capturing the motion of a galloping horse. **Left:** original movie. **Right:** original article in *Scientific American*.

Recent advances in 3D performance capture raised a large number of new problems. Among them, high level control of raw animated sequences is an important component of any editing or processing framework. While former, synthetic animated 3D sequences usually came with an underlying control structure (e.g., skeleton, cage) that tailored the animation at high level, 3D+time acquisition systems only provide raw point sets or mesh sequences, possibly consistent over time [22,70], but without any high level control structure. Such a structure is key for compression, motion editing and other processing tasks.

As already mentioned in the introduction of this thesis, high level motion control structures can be roughly classified in three categories. First, *skeletons* capture nicely locally rigid motions and are extensively used in synthetic human and animal body animation. Second, *surface handles* can be defined using a shape decomposition and help establishing a consistent segmentation of the model, allowing to morph independently each component w.r.t. rigging controllers. Third, *cages* are low resolution meshes which transfer their deformation to a high resolution model embedded in their encompassed space by the mean of a specific coordinate system. Recent advances in cage coordinate systems [46, 47, 61] now offer a very flexible framework to smoothly and efficiently edit the shape of models with arbitrary topology, without even being constrained manifold structure, using a simple cage. These three categories of structures have all their own strength and weakness, and rather complete each other than compete in a modelling and animation package.

To exploit their modelling power, with performance captured data for instance, a reverse engineering process is required to construct them automatically from a raw animated sequence. Most reverse engineering methods built on the same observation: a large part of the raw sequence motion can be captured at a coarser level and small local motions can be ignored in a number of application scenarii. Thus, the reverse engineering process consists in replacing a sequence of (high resolution) raw models by a single static one together with a sequence of (coarser) control structures. Then, the high resolution sequence can be reconstructed by applying the animated control structure sequence to the static model. This yields immediate benefits for compression and high level editing, but also for processing and analysis, as the coarse nature of the control structure makes it practical for a number of computationally expensive techniques.

Such a reverse engineering framework has been recently proposed for skeletons [23], where the resulting structure can be used for both processing and shape/motion modelling.

The case of cages however has not been entirely tackled, and existing methods fail to provide high quality animated cage sequences which are both able to reconstruct faithfully the input sequence and also well-structured enough to be exposed to the user for interactive post-editing.

In this chapter, we address this problem and present a new powerful framework for the compact and stable encoding of deforming 3D objects with cage-based representations. Given an animated sequence (with one-to-one vertex correspondences) along with an initial cage embedding, our technique generates smoothly varying cage embeddings which reconstruct the enclosed object deformation with controllable error. Our framework is constructed around an inversion process and exhibits the following features:

- **Generality:** our approach supports synthetic and captured sequences in the form of triangle soups, point clouds and volumetric data; it is oblivious to the cage coordinate system; it can implicitly deal with realistic (e.g., as-rigid-as-possible) as well as expressive (e.g., cartoon stretching) deformations. Thus, the resulting compact cage-based representation can act as an intermediate, low-memory footprint substitute to speed up time-varying geometry processing, while delegating the tricky task of detail preservation to the underlying cage coordinate model.

- **Speed:** high resolution animated sequences are usually heavy data sets; we propose an adaptive algorithm which reduces the number of constraints taken into account, and which maps naturally to GPUs.

- **Accuracy:** we measure the reconstruction error of our cage-based representation and show that it remains low and controllable, which allows its usage for compression.

- **Usability:** when exposed to the user, each reconstructed cage should be easy to edit ; we present an algorithm for localized cage regularization, yielding very smooth variations of the reconstructed cage over the sequence, while maintaining high accuracy in model reconstruction.

Technical contributions: In order to fulfill these criteria, we propose the following contributions:

- A fast and automatic framework for the extraction of *re-usable* deformation cages from arbitrary sequences of deforming 3D objects (section 8.3).

- A purely algebraic algorithm for the extraction of optimal geometry-sensitive handles for cage deformation based on *maximum-volume sub-matrices* [38], allowing for stable and fast cage coordinate inversions (section 8.4).

- A novel spectral regularization algorithm, enabling a localized enforcement of regularization terms, only for the cage vertices where the numerical solution of the inversion is found to be the most unstable (section 8.5).

We evaluate our framework on various synthetic and acquired data sets and demonstrate its usefulness for different applications (section 8.6):

- Compression: Since cages are meant to have much less vertices and faces that the enclosed model, they can be used for sequence compression. In that case, decompression consists in, given the enclosed model at the first frame only, reconstructing its poses along the sequence thanks to the cages generated by our algorithm. As these cages are smooth and obtained through stabilized inversion, they are not sensitive to *numerical instabilities* and guarantee stable sequence decompression. Also, in contrast to traditional compression strategies, this approach is oblivious to the representation of the enclosed model (non-manifold meshes, point clouds, etc.).

- Animation Transfer: Since they are stable and smooth, the cages generated by our algorithm can be used to enclose another object (in contrast to traditional inversion strategies), thus transferring the animation to the other object.

- Speeding up geometry processing tasks: Sharp features introduced by the artist in the "rest pose cage" are well-preserved in our cages, therefore they can be used as a low-resolution representation that is suited to heavy geometry processing tasks such as mesh interpolation.

8.2 Related work

Many deformation models have been recently proposed for interactive 3D shape modeling/editing, in particular using a variational framework [62,63,84] (see the complete survey by Botsch and Sorkine [15]). While control skeletons [42,97,103] are an industry standard to interact with the shape, volume deformations based on enclosing cages have become very popular due to their simplicity, flexibility and speed.

In particular, they are oblivious to the representation of the enclosed object (polygonal soup, point cloud, volumetric data etc.) and allow for efficient pre-computations, along with simple on-line computations based on linear combinations. This makes them a perfect support for deformation sequence encoding.

Cage-based deformation: As presented in the technical background section 3.3.2 of this thesis, a lot of work has been done on the definition of cage coordinates, allowing to transfer the deformation of a cage to its inner space.

Despite these advances in cage-based volume deformations, only little work payed attention to the stable computation of the inverse of the cage coordinates, a necessary step for the cage-based encoding of animated sequences.

Animation inverse kinematics: Several complete systems have been presented for the purpose of animation post-editing through the analysis of time-varying 3D shapes [53, 54, 87, 100]. However, their internal animation representation is not compliant with

industry standards (e.g., skeletons or cages), which reduces their applicative impact, as discussed in [23].

More specialized techniques specifically address the extraction of intermediate, manipulable shape representations for the purpose of easy post-editing, such as reduced deformation models [24,56] or control skeletons [80]. However, they do not encode the deformation itself, which restricts their application to pose post-editing only. In contrast, de Aguiar et al. [23] introduced the first framework able to fully convert the animation into a compact and editable representation. Their algorithm reconstructs a plausible skeleton corresponding to the input animation as well as the skeleton's deformation parameters enabling a faithful reconstruction of the input sequence, useful for compression for instance. However, such techniques are mostly restricted to rigid motion estimation, suitable for human bodies for instance, but not for sequences subject to stretching or volume variation. More importantly, such methods are exclusively surface-based and require a manifold input. Also, skeletons are mostly suited for shapes exhibiting tubular components and they are difficult to extend to arbitrary shapes. A few skinning approaches [43,51] allow to reverse-engineer an animated mesh by considering moving 3D frames instead of the skeleton's bones. This flexibility allows to process sequences including non-trivial motion, like cloth motion for instance [51]. However, the absence of a global structure on these *frames* limits the spectrum of their applications to compression.

Only a few approaches address the stable and efficient inversion of cage-based deformations. Xu et al. [100] propose to transfer an initial cage embedding from one frame of the sequence to the others. However, their algorithm does not explicitly invert the transformation, but instead tries to compensate the cage pose change through local rotation blendings, which turns to be inaccurate and unstable, especially when the cage is not close to the object. Similarly, a possible strategy could consist in transferring the motion from the mesh to the cage, using deformation transfer algorithms [88]. However, this strategy would induce a large reconstruction error of the model by the deformed cage since there is no guarantee in general that the cage reproduces in its interior the transformation it undergoes. For instance, an as-rigid-as-possible (ARAP) transformation of the cage does not induce an ARAP transformation of the enclosed model and reciprocally. The same remark goes for any direct spatial encoding of the cage w.r.t. the model geometry, such as maintaining the cage as an offset of the animated mesh for example. Moreover, the inversion of the cage *must* take into account the coordinate system used for reconstruction. Indeed, for a given cage, reconstructing an enclosed object with different coordinates (Mean Value Coordinates, Green Coordinates, Harmonic Coordinates, Positive Mean Value Coordinates, Maximum Entropy Coordinates, etc.) leads to different results.

Ben-Chen et al. [7,8] automatically compute new cage embeddings to satisfy sparse user constraints through a sequence of least squares fitting. However, their framework is using GC only and enforces pure rotations on the *medial axis* of the shape, as well as Hessian's norm minimization on the cage, and is highly restricted to as-rigid-as-possible deformations and does not specifically address general and stable cage inversion.

Savoye et al. [79] introduce a global regularization term aiming at preserving the differential coordinates of the cage vertices. Being not rotation invariant, such an approach leads to shrinking artifacts and fails at preserving the cage geometry, which is often carefully designed by artists and critical for editing. Finding an efficient but general regularization

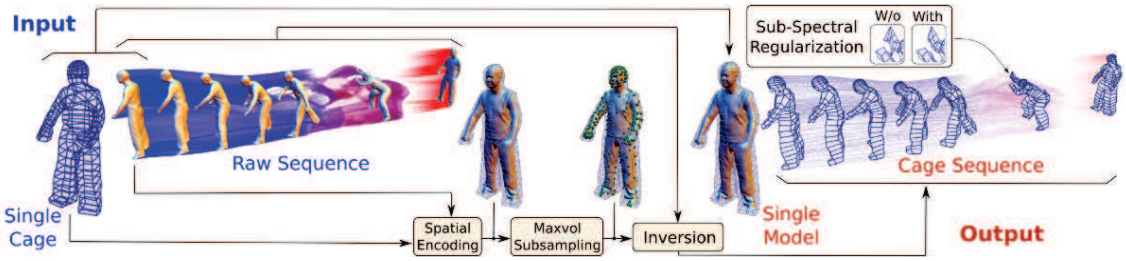


Figure 8.2: **Processing Pipeline.** From left to right: Given a raw 3D+t sequence and an initial cage, we first extract an optimal subset of positional constraints for the cage coordinate inversion. Then, the cage coordinates are inverted for each frame of the input sequence. A selective enforcement of arbitrary regularization terms is defined to affect the cage vertices where the inversion is the most unstable. The resulting smoothly varying cage sequence faithfully reconstructs the input sequence.

term, with no a priori on the input sequence, and which does not completely smooth away the cage geometry, turns out to be unexpectedly difficult. We cope with it by reshaping the linear least squares fitting pipeline with a novel, efficient inversion algorithm, coupled with a new selective regularization scheme, which only regularizes the cage vertices where the inversion is less numerically stable.

8.3 Overview

Input: We consider a raw animated sequence of N frames, represented by a set of points $\mathcal{P}$ with time-varying embeddings in $\mathbb{R}^3$ $(\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_N)$. This position data can correspond to the embedding of a triangle mesh, a point cloud or a volumetric mesh. Additionally, we consider an input cage $\mathcal{C}_R$ enclosing an arbitrary reference frame $\mathcal{P}_R$ of the sequence.

Inversion procedure: Our processing pipeline (Fig. 8.2) is composed of three major stages:

1. Compute the spatial encoding Φ of $\mathcal{P}_R$ into $\mathcal{C}_R$ using a cage coordinate system (e.g., MVC, HC, GC).

2. Identify the set of handles $\mathcal{H}_R \subset \mathcal{P}_R$ which maximizes the volume of a squared version of the cage coordinate matrix. This property stabilizes and speedups the inversion while guaranteeing dense inversion spectra.

3. For each frame $\mathcal{P}_t$, the corresponding cage embedding is computed by inverting the cage coordinates, using $\mathcal{H}_t$ (embedding of $\mathcal{H}_R$ at frame t) as positional constraints.

Additionally, our framework can selectively apply arbitrary regularization terms on the highest frequencies of the spectrum of the inversion (the last singular vectors of the coordinate matrix singular value decomposition).

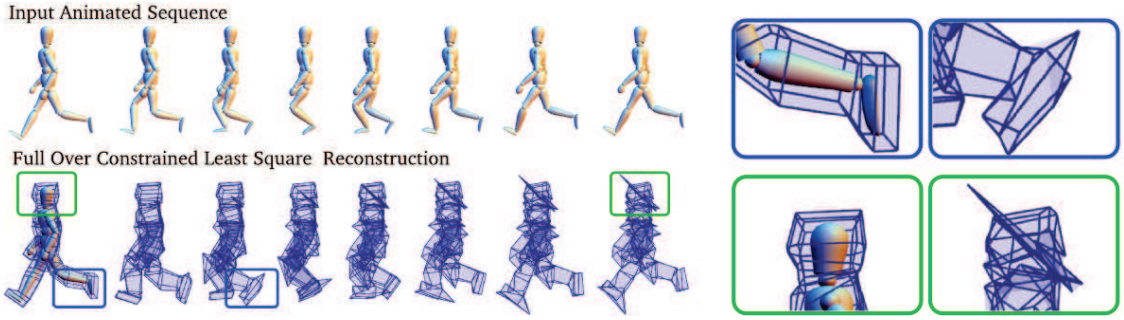


Figure 8.3: Cage coordinate (MVC) inversion using a traditional over-constrained least-squares fitting results in unstable cage inversions (middle row). The output cages exhibit spikes and other artifacts (bottom row), which makes them unusable for manual post-editing or compression.

Output: The resulting set of smoothly varying cages compactly and faithfully reconstructs the input sequence.

8.4 MaxVol based cage inversion

In the following, we present our algorithm for the stable inversion of the cage coordinates. We briefly discuss why a simple strategy based on over-constrained least-squares fitting can be unsatisfactory. Then we present our algebraic algorithm based on optimal handle identification.

8.4.1 Problem statement

Given an enclosed 3D model represented by its reference position data $\mathcal{P}_R$ and a closed triangle cage mesh $\mathcal{C}_R$, cage coordinates (such as MVC [47], HC [46] or GC [61]) allow to encode each point position $p_i \in \mathbb{R}^3$ of $\mathcal{P}_R$ w.r.t. to the cage vertex positions $c_j \in \mathbb{R}^3$ (and triangle normals n_j for GC) of $\mathcal{C}_R$ by: $p_i = \sum_j \phi_j(i) \cdot c_j$, or $\mathcal{P}_R = \Phi \cdot \mathcal{C}_R$, where $\mathcal{P}_R$ is represented as an $(n \times 3)$-matrix, Φ is a rectangular $(n \times m)$-matrix and $\mathcal{C}_R$ is a $(m \times 3)$-matrix.

Given a set of poses $\mathcal{P}_t$ of the model, we aim at computing a set of corresponding cages $\mathcal{C}_t$, such that $\Phi \cdot \mathcal{C}_t \simeq \mathcal{P}_t$. The L_2-projection of $\mathcal{P}_t$ onto the space of admissible deformations is $\overline{\mathcal{P}_t} = \Phi \cdot \Phi^\dagger \mathcal{P}_t$, which involves the pseudo-inverse $\Phi^\dagger$ of Φ. As the cage coordinate matrix Φ is large and dense, the computation of its pseudo-inverse $\Phi^\dagger$ (an $(m \times n)$-matrix) through singular value decomposition (SVD) is expensive. A faster approach consists in solving the equivalent system, where $(\Phi^T \cdot \Phi)^\dagger$ is an $(m \times m)$-matrix:

$$\mathcal{C}_t = (\Phi^T \cdot \Phi)^\dagger \Phi^T \mathcal{P}_t \tag{8.1}$$

This yields an over-constrained linear system, where the number of unknowns m (the number of vertices in C_t for MVC, plus the triangle normals for GC) is meant, by definition of control cages, to be significantly smaller than the number of constraints n (the number

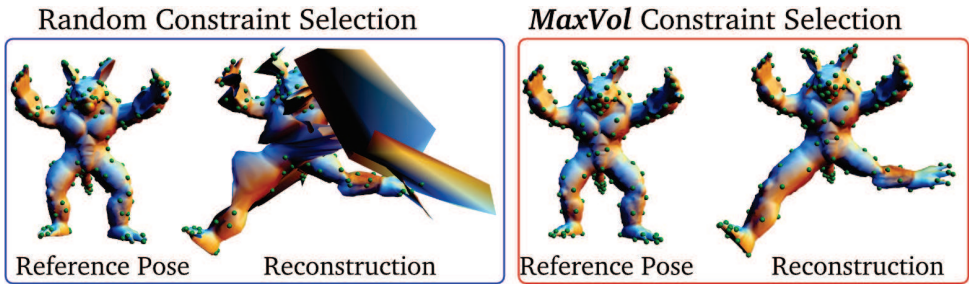


Figure 8.4: Minimal selection of position constraints (as many as cage vertices, MVC). Selecting constraints randomly on the reference frame $\mathcal{P}_R$ (left) generates a cage for the other frames which performs poor reconstruction (middle left). In contrast, our relaxation strategy (with the same amount of constraints) generates a smooth reconstruction (right).

of point samples in the model). Let $U\Sigma V^T$ be the SVD of $(\Phi^T \cdot \Phi)$. The solution is given by:

$$c_{j_t} = \sum_k V_{jk} \cdot \left((U^k)^T \cdot \Phi^T \cdot \mathcal{P}_t \right) \cdot \frac{1}{s_k} \tag{8.2}$$

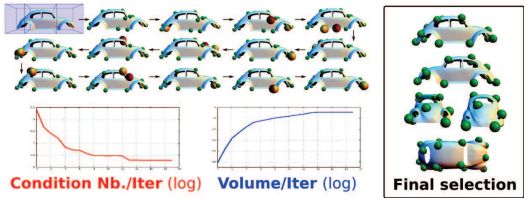
with s_k the k^{th} singular value.

However, a slight perturbation $\delta_{\mathcal{P}_t}$ on the constraints $\mathcal{P}_t$ may be drastically amplified through the SVD when projecting onto the singular vectors associated with low singular values (Eq. 8.2). We illustrate this phenomenon in Fig. 8.3: over-constrained least-squares fitting yields instabilities resulting in important spikes on the output cages, which makes them useless application-wise. To overcome this issue, we first propose a strategy to optimally relax the system, which has the beneficial side-effect of decreasing the condition number of the coordinate matrix, hence stabilizing its inversion.

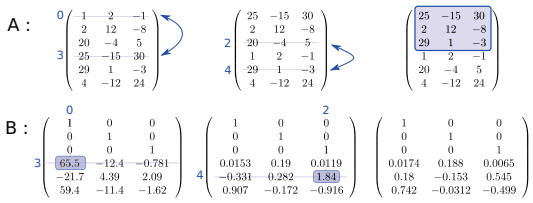
8.4.2 MaxVol relaxation

Reducing the number of constraints of an over-determined linear system reduces the chances of taking into account multiple conflicting constraints. Traditionally, an arbitrary subset of the constraints is considered, whose size is progressively reduced until a satisfactory trade-off between stability and precision is obtained. Ultimately, one could narrow the size of the constraint set down to that of the unknowns. However, as shown in Fig. 8.4, this selection process must be carefully carried out to maintain a decent solution precision.

Minimum condition number sub-matrix To optimally reduce the number of constraints, while guaranteeing a stable inversion process, one needs to seek the optimal square $(m \times m)$ sub-matrix $\Phi_\square$ of Φ, with minimum condition number. This linear algebra problem is well known to be NP-hard. As discussed by several authors [18,31,38], computing ϵ-approximations is NP-hard as well.



(a) Each iteration of the maxvol algorithm (top) corresponds to deselecting a constraint (red), and selecting another (yellow). The condition number and the volume of $\Phi_\square$ are respectively shown in red and blue (log scale).



(b) Iterations of the *maxvol* algorithm on a toy example. At each iteration, the location of the element B_{ij} with maximum absolute value indicates which rows to flip in A to improve the volume of $A_\square$ (in blue on the right).

Figure 8.5: MaxVol illustrated on a toy example

Maximum volume sub-matrix A weaker indication of the *invertibility* of a matrix is given by the value of its determinant, which ought to be high for the matrix to admit stable inversion. Finding a square sub-matrix that has maximum volume (absolute value of the determinant) has been intensively researched by the linear algebra community and an efficient iterative algorithm has been proposed recently [38]. Although it is not guaranteed to identify the optimal sub-matrix maximizing its volume, its practical performances demonstrate significant decrease of the condition number, which in the worst case falls back to that of the over-determined system. In the following, we sketch the main steps of this algorithm and detail its integration into our framework.

MaxVol algorithm Let A be an $(n \times m)$-matrix $(m < n)$. The goal of the algorithm is to identify the m rows of A such that the resulting square $(m \times m)$ sub-matrix has maximum volume. The algorithm *MaxVol* is based on simple observations, further discussed in the original paper [38].

Let $A_\square$ be the top square sub-matrix of A, composed of the first m rows of A, and $B = A \cdot A_\square^{-1}$. Swapping the rows $i > m$ and j in A for which B_{ij} has maximum absolute value increases the volume of the top square sub-matrix of A, if $|B_{ij}| > 1$. Let e_j be the column vector with value 1 at j and 0 elsewhere. Then, both swapping the rows i and j in A and updating B can be performed in a single update of the form:

$$B := B - (B_{:j} - e_j + e_i) \cdot (B_{i:} - e_j^T)/B_{ij} \tag{8.3}$$

where $B_{:j}$ depicts the entire j^{th} column of B and $B_{i:}$ its entire i^{th} row.

This yields a practical iterative algorithm (Fig. 8.5(b)):

- **Initialization:** Order the rows of A, such that its top square sub-matrix $A_\square$ is invertible. Compute $B = A \cdot A_\square^{-1}$.
- **Iteration:** Find the maximum absolute value element B_{ij} with $i > m$. Swap rows i and j in the current solution. Update B with Eq. 8.3.

The iterations can be stopped on demand or when the maximum B_{ij} gets smaller than one. In terms of complexity, the initialization stage takes $O(m^2 \times n)$ operations and each iteration takes $O(m \times n)$ steps.

8.4.3 Cage inversion based on maxvol relaxation

Applying the maxvol algorithm to our setting is straightforward, the input being Φ and the output being $\Phi_\square$. In practice, we observed that about $2 \times m$ iterations are required until convergence. Fig. 8.5(a) illustrates a few iterations of the algorithm. Both the volume and the condition number of $\Phi_\square$ respectively increases/decreases at each iteration (results are shown in log scale), assessing the *invertibility* quality of $\Phi_\square$.

Cage coordinate inversion For each frame $\mathcal{P}_t$, the corresponding cage embedding $\mathcal{C}_t$ is given by (where $\mathcal{H}_t$ is the sub-matrix of the top m rows of $\mathcal{P}_t$):

$$\mathcal{C}_t := \Phi_\square^\dagger \mathcal{H}_t \tag{8.4}$$

The pseudo-inverse $\Phi_\square^\dagger$ is computed through SVD, as discussed in the case of $(\Phi^T \cdot \Phi)^\dagger$ in Eq. 8.2 ($\Phi^T \cdot \mathcal{P}_t$ needs to be exchanged with $\mathcal{H}_t$). Notice that for each frame, the rows of $\mathcal{C}_t$ and $\mathcal{P}_t$ also need to be swapped according to the swaps performed in Φ by maxvol. Fig. 8.10 provides a comparison of the repartition of the singular values yielding from the SVD, between our technique ($\Phi_\square$) and the classical over-determined approaches (Φ and $\Phi^T \cdot \Phi$). Notice that $\Phi_\square$ exhibits a more *convex* spectrum. Also, as discussed with Eq. 8.2, important instability occurs in the inversion if the last singular values of the SVD are low. With our relaxation scheme, the last singular values of $\Phi_\square$ are much higher than those of the traditional over-determined approximations (Fig. 8.10, rightmost plots), which yields much more stable inversions.

Finally, once $\Phi_\square$ is computed, the inversion of the cage embeddings for the whole sequence is faster with our approach than with the traditional over-determined least-square strategy. In both cases, the SVD of an $(m \times m)$ matrix needs to be computed in a pre-process. For each frame, with our strategy, the solution is obtained by the multiplication of an $(m \times m)$ by an $(m \times 3)$-matrix (Eq. 8.4). For the traditional least-squares approach however, an extra multiplication between an $(m \times n)$ and an $(n \times 3)$-matrix is required (Eq. 8.1).

Geometric interpretation Although our strategy is purely algebraic, it yields interesting geometrical insights. When reflected on the matrix $\mathcal{P}_R$, the row swaps of Φ are equivalent to deselecting a constraint and picking another one instead (Fig. 8.5(a)). Near convergence, the constraints to flip get closer and closer. At the end of the process, the set of corresponding points of $\mathcal{P}_R$ (in green in Fig. 8.5(a)) can be interpreted as a minimally optimal set (noted $\mathcal{H}_R \subset \mathcal{P}_R$) of handles for deformation, given the model $\mathcal{P}_R$, the cage $\mathcal{C}_R$ and the employed cage coordinate model. As shown in Fig. 8.6, the local density of $\mathcal{C}_R$ is captured by $\mathcal{H}_R$. Also, configurations implying high curvature, sharp edges, or boundaries (Fig. 8.5(a)) are captured by $\mathcal{H}_R$, while the symmetries of the cage are also represented by $\mathcal{H}_R$. In other words, the constraints $\mathcal{H}_R$ are well dispatched in the space of cage coordinates.

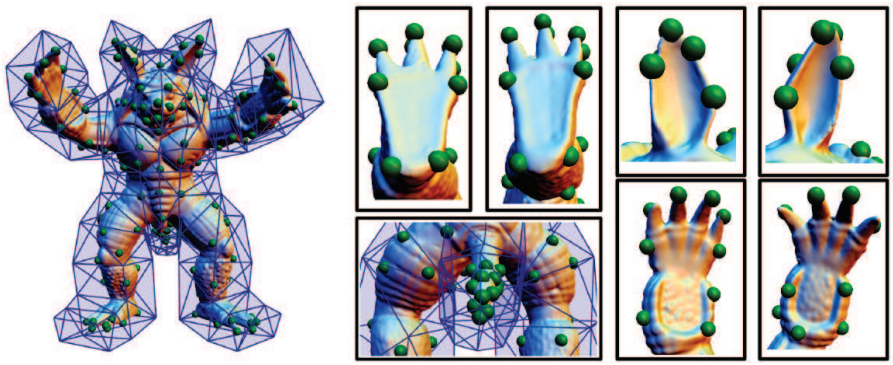


Figure 8.6: Geometry-sensitive deformation handles extracted by our **algebraic-only** algorithm. Sharp edges (ears) and high curvatures (finger tips) are implicitly detected as constraints. The sampling density (tail) and the symmetry of the cage are also captured by the constraints.

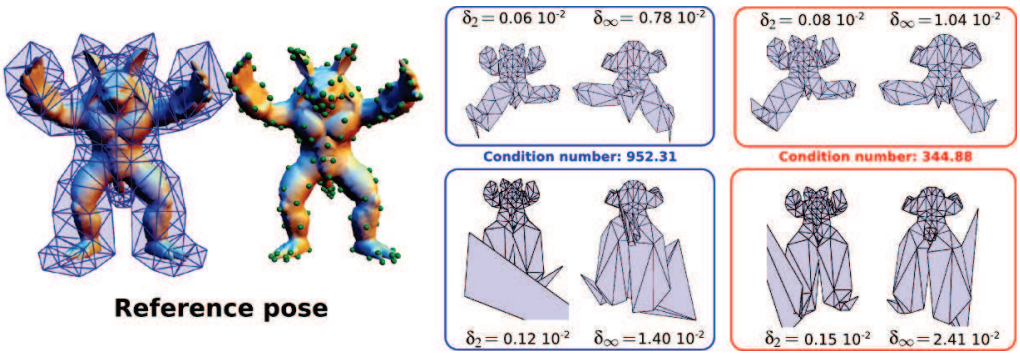


Figure 8.7: Comparison between the over-determined least-squares approach (blue) and our relaxation strategy (red) for deformations based on rotations (top) and stretching (bottom). The point-to-point L_2 and L_∞ distances between the reconstructed model and $\mathcal{P}_t$ are given by δ_2 and δ_∞ (fraction of the model bounding box diagonal).

Fig. 8.7 shows the cage reconstruction with MVC of two poses with our inversion strategy, compared to the classical over-determined least-squares. Since our strategy relaxes the set of constraints, the error in the *model* reconstruction is necessarily slightly higher. However, as shown on the right, our approach removes many of the *cage* instabilities occurring with the over-determined approach.

Coordinate system analysis Fig. 8.8 presents the spectral behavior of the different cage coordinate systems used in this chapter, namely MVC, HC and GC. MVC and HC exhibit similar spectral behavior: in comparison to the over-determined full least-squares system (Fig. 8.8, bottom), MaxVol exhibits much higher last singular values, which stabilizes the inversion. In the case of GC, MaxVol increases the singular values all over the spectrum. In all the cases, MaxVol decreases significantly the condition number of the system.

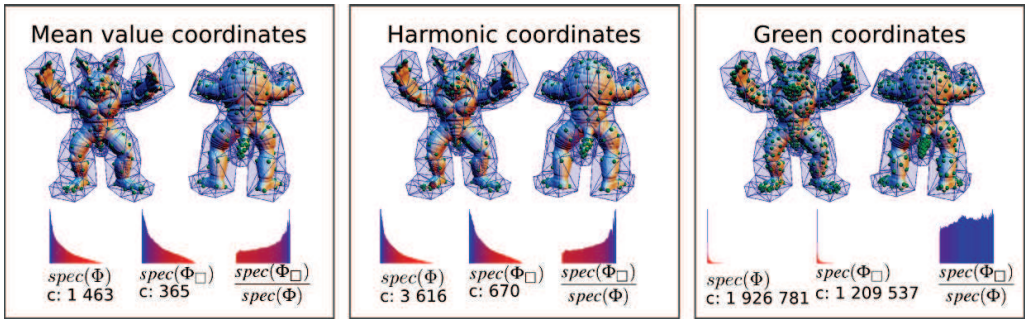


Figure 8.8: MaxVol handles and coordinate matrix spectra for different coordinate systems (MVC, HC, GC). In each case, MaxVol drastically reduces the condition number of the system ($\Phi_\square$, middle) in comparison to over-determined least-squares (Φ, top). The gain (bottom row) is maximum in high frequencies, where the inversion is the most instable.

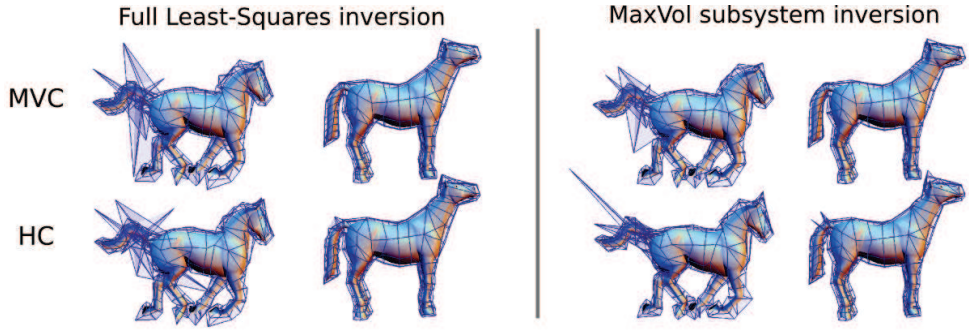


Figure 8.9: Comparison between over-determined and MaxVol inversions with different coordinate systems (mean value coordinates and harmonic coordinates).

Fig. 8.9 shows the cage reconstruction with our inversion strategy (MaxVol or not), with different coordinate systems (MVC, HC). Since they do not admit a close form expression, HC need to be *approximated* with a solver, which necessarily leads to *residuals* (as disccused in [46]), whose importance is implementation-dependant. This imprecision can be observed when inverting the reference pose $\mathcal{P}_R$ (Fig. 8.9, right): the inversion does not result in the identity (spike on the base of the tail).

The GC are the most instable to invert. The space of transformations they allow is too large to allow accurate inversions if the face normals are not constrained to be orthogonal to the cage faces. Also, the inversion uniformly processes all the unknowns of the system whereas for GC, these mix much different entities (vertex positions *and* face normals). As a result, the geometry of the cage is globally unstable.

As shown in Fig. 8.7, our relaxation strategy significantly reduces the condition number of the system, resulting in less instabilities on the inverted cages. Still, a few of them may remain (Fig. 8.7, right). To overcome this issue, we introduce a novel regularization strategy, that selectively focuses on the unknowns for which the most instabilities occur.

8.5 Sub-spectral regularization

Traditionally, to cope with the instability of a linear system, a solution consists in introducing *globally* a *geometrical* regularization term. This enables to bias the solution towards a space of preferred solutions (e.g. detail-preserving or as-rigid-as-possible transformations). However, global regularization also comes with several drawbacks. First, such a solution is not general, since it makes assumptions on the transformations present in the input sequence. Second, even with a priori information on the model transformations, there is no direct connection to the transformations that the cage should undergo. Third, in practice, this solution tends to over-damage the data fitting, even before all of the instability is corrected.

8.5.1 Regularization terms

In our setting, without any a priori on the space of transformations exhibited in the input sequence, one of the only desirable properties of a regularization term is its invariance against rotation. For instance, Savoye et al. [79] enforce the preservation of the differential coordinates of the cage vertices. However, this is not rotation invariant and leads to significant shrinking artifacts. The preservation of the norm of the Laplacian of the cage vertices does not suffer from this drawback but involves non linear terms [30].

Minimizing the Laplacian of the cage vertex positions is rotation invariant but, without sophisticated reconstruction strategy [12], it smooths away the cage curvature. The same holds for the minimization of the deformation Hessian [8, 90]. In other words, rotation invariant linear regularization terms are *destructive*, in the sense that they will smooth away the information provided in the input cage $\mathcal{C}_R$. In the following, we present a new regularization algorithm that locally allows the usage of *destructive* terms only for the unknowns where the inversion solution is unstable and not reliable.

8.5.2 Sub-spectral regularization algorithm

Let $U\Sigma V^T$ be the SVD of $\Phi_\square$. The vector of the cage vertex positions (and of its face normals for GC) can be expressed as a linear combination of the columns of V: $\mathcal{C}_t = \sum_k V^k \cdot q_{k_t},\ q_{k_t} \in \mathbb{R}^3$. The solution to the system inversion is given with $q_{k_t} = (U^k)^T \cdot \mathcal{H}_t / s_k$. A common strategy for sub-spectral regularization [39] consists in directly cropping the spectrum by setting the inverse of the last singular values to 0. Instead, a second common solution [39] consists in parameterizing the stability of the solution by a scalar α, and using s'_k instead of s_k, where $s'_k = \frac{s_k}{s_k^2+\alpha^2}$. These two strategies do not allow for the insertion of a geometrical regularization term, which would take into account the physics of the problem. In the following, we introduce a novel and general sub-spectral algorithm, which allows to include relevant geometrical regularization terms only on the most instable portions of the spectrum of $\Phi_\square$.

As shown in Fig. 8.10, the left part of the spectrum of $\Phi_\square$ can be interpreted as the low frequencies of the inversion, while the right part can be interpreted as its high frequencies. Also, as discussed earlier, the *cage* reconstruction instability is more likely to occur on the high frequencies of the spectrum, where the singular values are low. The last singular vectors of $\Phi_\square$ correspond to its *"pseudo-kernel"*, in the sense that a perturbation in that space induces only a slight change in the *model* reconstruction, since it is amplified by the corresponding singular value. In other words, relaxing the data-fitting constraints for these vectors will have a negligible impact on the reconstruction of the model. Therefore, our technique enforces regularization terms (expressed through a matrix Δ) only on the high frequencies of the spectrum. As the last singular vectors are highly localized (see Fig. 8.11), this guarantees a minor *destructive* impact of the regularization term on the global aspect of the cage inversion (its low frequency component) while locally fixing instabilities.

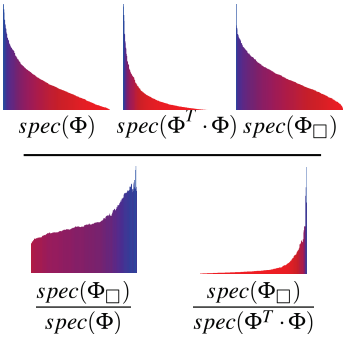


Figure 8.10: Comparison of the coordinate matrix spectra for the sequence of Fig. 8.2 with MVC (normalized plots).

Given a *spectrum threshold* s, the low frequency part of the spectrum is directly reconstructed from the positional constraints $\mathcal{H}_t$:

$$q_{k_t} = (U^k)^T \cdot \mathcal{H}_t / s_k, \qquad \forall k < (m - s) \tag{8.5}$$

The high frequency part is reconstructed with the traditional combination of positional constraints and regularization, to minimize the following energy ($\lambda \in [0, 1]$):

$$E = \lambda ||\Delta \cdot \mathcal{C}_t||^2 + ||(\Phi_\square \cdot \mathcal{C}_t) - \mathcal{H}_t||^2 \tag{8.6}$$

which is equivalent to (with q_{k_t} fixed $\forall k < (m - s)$):

$$E = \lambda || \sum_{k<(m-s)} \Delta \cdot V^k \cdot q_{k_t} + \sum_{k\geq(m-s)} \Delta \cdot V^k \cdot q_{k_t} ||^2 + \quad ||(\Phi_\square \cdot \mathcal{C}_t) - \mathcal{H}_t||^2 \tag{8.7}$$

Minimizing equation 8.7 is equivalent to solving the following linear system in the least squares sense, where the unknowns are q_{k_t}, $\forall k \geq (m - s)$ and where q_{k_t} are fixed $\forall k < (m - s)$ (cf Eq. 8.5):

$$\begin{cases} \sqrt{\lambda} \cdot \sum_{k\geq(m-s)} \Delta \cdot V^k \cdot q_{k_t} = -\sqrt{\lambda} \cdot \sum_{k<(m-s)} \Delta \cdot V^k \cdot q_{k_t} \\ s_k \cdot q_{k_t} = s_k \cdot \left((U^k)^T \cdot \mathcal{H}_t / s_k\right), \quad \forall k \geq (m - s) \end{cases} \tag{8.8}$$

The first line of Eq. 8.8 is the regularization energy expressed on the last singular vectors of the SVD, and the second line of Eq. 8.8 is the data fitting energy. This strategy can be used with arbitrary linear regularization terms expressed through a matrix Δ. Figs. 8.12 and 8.13 show examples of cage and model reconstructions with a particularly destructive term, the minimization of the cage Laplacian. As shown in Fig. 8.12, our spectral strategy enables a progressively localized blending of the data-fitting constraints

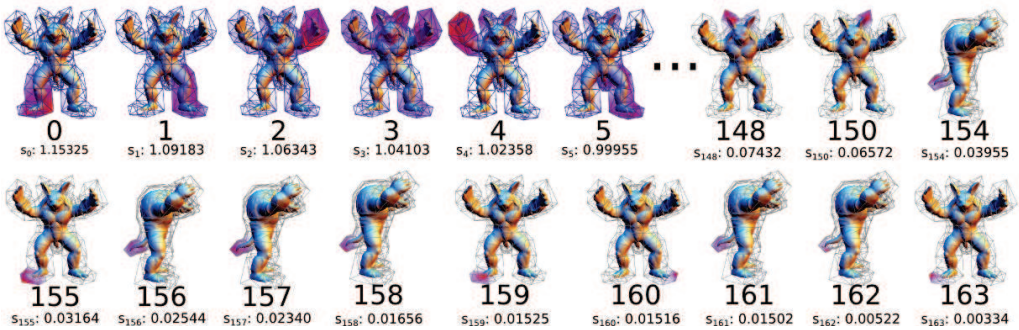


Figure 8.11: Singular vectors of $\Phi_\square$ on the cage (MVC, in absolute value, gradient of pink). For the last singular values, where instabilities may occur, these vectors are localized to a small subset of cage vertices.

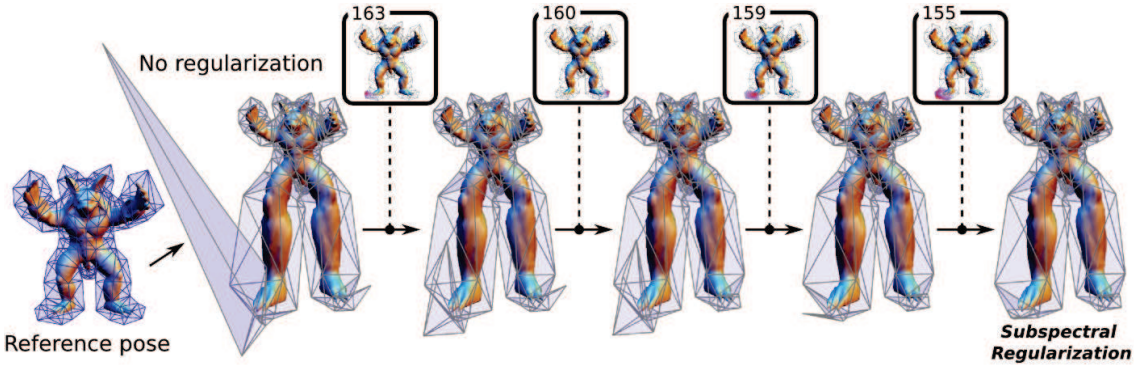


Figure 8.12: Cage and model reconstruction with MVC and Laplacian minimization. Increasing the *spectrum threshold* yields localized and progressive blending of the unstable cage vertex solutions with the solutions of the regularization.

with the regularization term, while the rest of the cage remains unaffected. Fig. 8.13 provides further comparisons between a classical regularization strategy and our spectral approach on a challenging cage with sharp dihedral angles and a deformation with high distortion. The color map plots the point-to-point distance with the non-regularized cage. In other words, it depicts the local quality of the data fitting in term of *cage* reconstruction. Whereas classical regularization damages globally the reconstructed cage, our strategy automatically localizes the effect only on the unstable cage vertices, while preserving the *true* solution on stable ones, yielding much more accurate *model* reconstruction.

8.6 Results

In this section, we present the results of our technique on data-sets coming from spatio-temporally coherent motion captures [22] and synthetic animations. These experiments have been run under Linux on a commodity laptop with a Core 2 Duo 2.53 GHz CPU and a GeForce GTX 260 M GPU. Our programs are written in C++. The maxvol algorithm

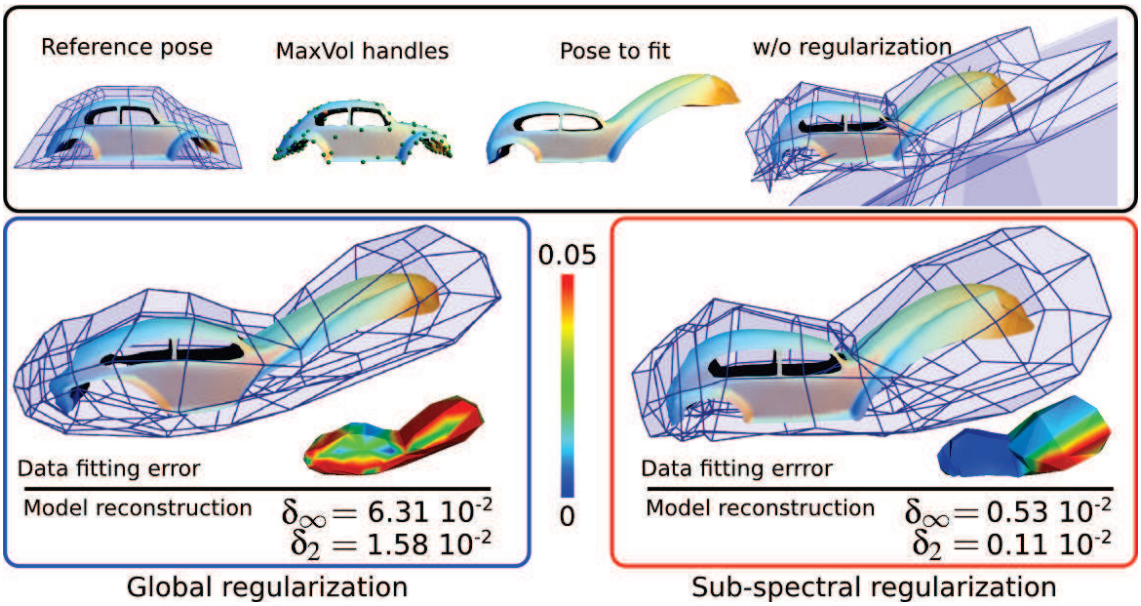


Figure 8.13: Localized effect of our sub-spectral regularization ($s = 20$), in comparison to traditional global regularization, on a *destructive* regularization term: the Laplacian minimization (MVC and maxvol relaxation). In both cases, the regularization weight λ is set to the same value (1).

is fully implemented on the GPU with CUBLAS, while SVD computation uses GSL and OpenNL is used to solve Eq. 8.8. Unless explicitly mentioned, all the results have been computed with the same default setting (MVC, MaxVol selection, Laplacian minimization, $\lambda = 1$, $s = 20$). Distance measurements are expressed w.r.t. the bounding box diagonal.

8.6.1 Encoding quality

Fig. 8.14 shows the cage-based encoding of several input sequences, along with the resulting reconstruction. The corresponding timings are provided in table 8.1. Note, that the cage embeddings smoothly vary over time, while guaranteeing low model reconstruction error. Hence, the output cages can be used for post-editing or compression. The algorithm, without any geometrical a priori, implicitly handles acquired data-sets (top-4 rows) and synthetic data motioned through as-rigid-as-possible (bottom row) or cartoon-like stretching deformations (Fig. 8.12). The algorithm successfully reconstructs the sequence despite chaotic deformations (cf. the skirt motion, 3rd row). Note, that in this specific example, the spectral regularization automatically removed unstable cage spikes but preserved those which are mandatory for a correct reconstruction.

As demonstrated in Fig. 8.15, removing systematically all the spikes appearing on the skirt would require a regularization up to the 50^{th} last singular vectors. However, the corresponding singular value is then more than 0.1, which would induce significant model reconstruction errors. Then, the remaining spikes appearing in Fig. 8.14 are necessary

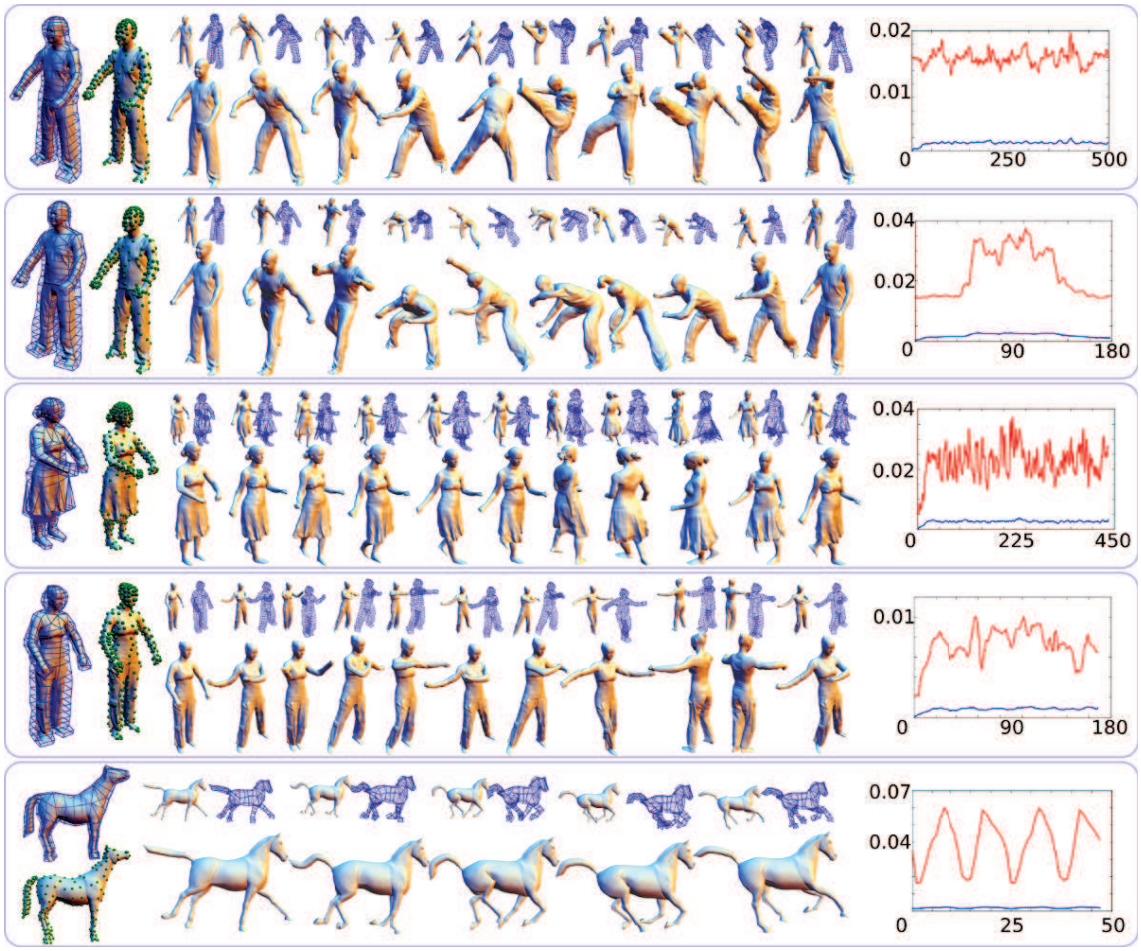


Figure 8.14: **Reconstructed sequences:** for each sequence, $\mathcal{P}_R$, $\mathcal{C}_R$ and $\mathcal{H}_R$ are shown on the left. For each frame t, on the top, $\mathcal{P}_t$ appears on the left; the reconstructed cage is on the right, the reconstructed frame is at the bottom (larger view). The blue and red curves respectively show the evolution of the point-to-point L_2 and L_∞ model reconstruction error.

for an accurate model reconstruction. Moreover, these features of the cage still evolve smoothly along the sequence.

8.6.2 Encoding robustness

Fig. 8.17 shows a reconstruction example with random perturbations inserted in the sequence to fit (each vertex is displaced randomly within a radius equal to 0.3×10^{-2} times the bounding box diagonal). The algorithm still generates a valid reconstruction. Note that the curves of the model reconstruction errors exhibit the same overall behavior than those of the original data-set (Fig. 8.14, top row) but with additional noise. In particular, the difference between the maximum L_∞ errors is 0.36×10^{-2}, which exactly reflects the introduced noise and which further assesses the robustness of the algorithm.

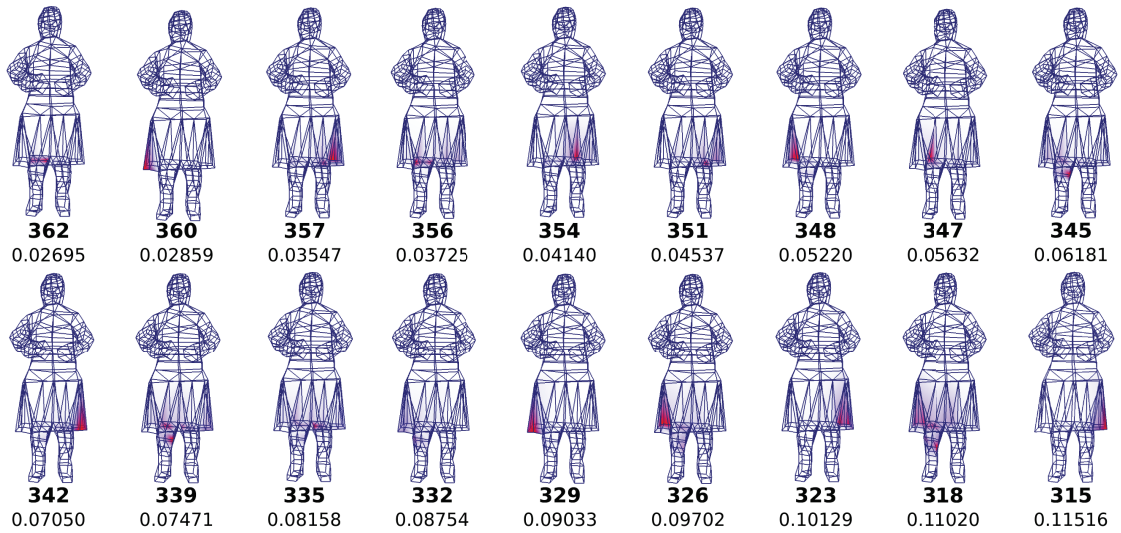


Figure 8.15: Singular vectors (with their singular values) localized on the skirt of the *Skirt* model. A change in the 315^{th} singular vector induces a change in the model reconstruction, up to a factor 1/10.

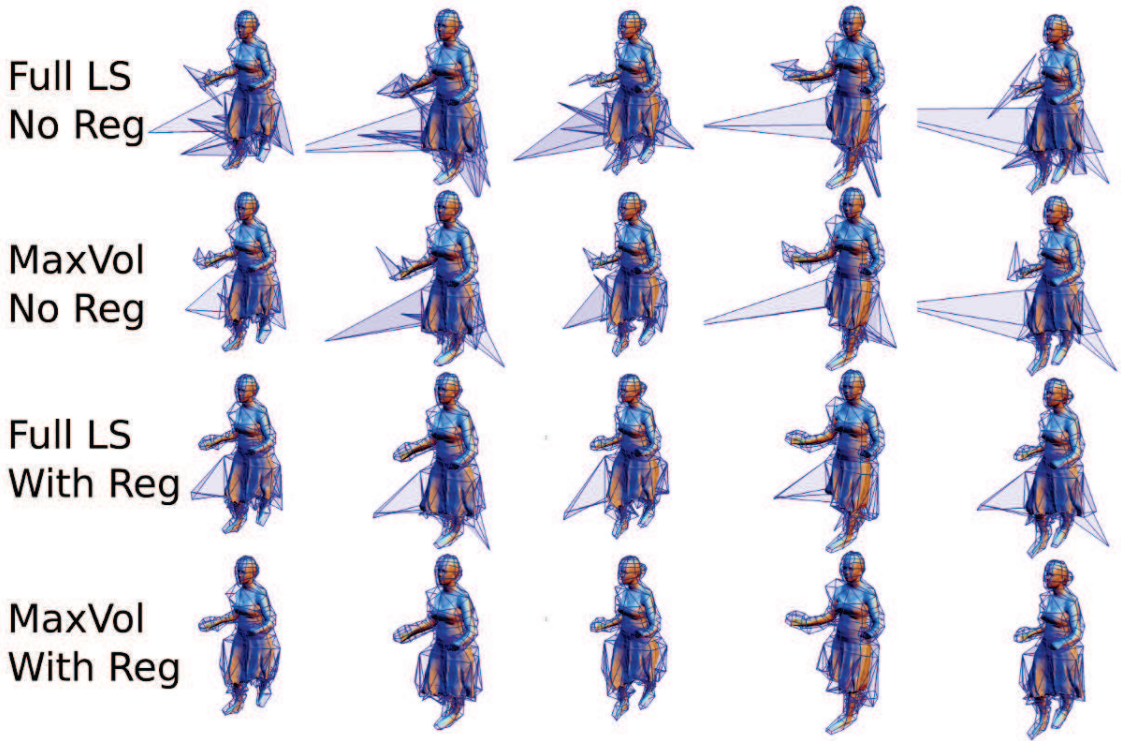


Figure 8.16: Relative importance of the individual steps of our approach: inversion of the *Skirt* sequence with the over-determined least-squares (*Full LS No Reg*, first row), MaxVol (*MaxVol No Reg*, second row), the over-determined least-squares with spectral regularization (*Full LS With Reg*, third row, $s = 50$), MaxVol with spectral regularization (*MaxVol With Reg*, fourth row, $s = 50$).

Figure 8.17: Reconstruction (right) of a noisy sequence (left).

Figure 8.18: Encoding with a reference cage **generated automatically**. The surface is first voxelized. We then contour a slight offset of this volume before simplyfing it using QSlim.

Input Sequence	Points	Cage (#V / #T)	Frames	Pre-process: MaxVol + SVD	Av. Frame Inv.	Total (ms)
Fig. 8.14, 1st row	19,998	330 / 656	**499**	5,616	568	**289,048**
Fig. 8.14, 2nd row	19,998	330 / 656	**179**	5,616	568	**107,288**
Fig. 8.14, 3rd row	19,990	368 / 732	**437**	7,291	890	**396,221**
Fig. 8.14, 4th row	15,002	339 / 674	**169**	5,617	624	**111,073**
Fig. 8.14, 5th row	8,431	348 / 692	**48**	5,029	748	**40,933**

Table 8.1: Detailed running times for Fig. 8.14. The pre-process, dominated by MaxVol, and the frame inversion timings are mostly dependent on the number of unknowns (number of cage vertices for MVC). All timings are expressed in ms.

Fig. 8.18 shows a reconstruction with a reference cage computed **automatically** (602 vertices). Interestingly, the reconstruction error is lower (with an average L_∞ error of 0.73×10^{-2} against 1.39×10^{-2} in Fig. 8.14, top row). Indeed, realistic cages designed by artists (Fig. 8.14) are more challenging to fit, given their specific structure and the very small number of unknowns they provide to the inverse system.

8.6.3 Comparisons and limitations

The localized nature of our sub-spectral regularization allows to reconstruct a cage sequence which is more faithful to the input reference cage, avoiding the rounding artifacts stemming from *destructive* regularization terms (Fig. 8.13). Fig. 8.19 provides a comparison between our algorithm and a traditional, globally regularized, over-determined least-squares (with the exact same setting). Important artifacts occur with the traditional approach (see the inset zooms), yielding higher reconstruction error (with an average L_2 error of 0.24×10^{-2} against 0.14×10^{-2} with our approach). As discussed earlier, not only our sub-

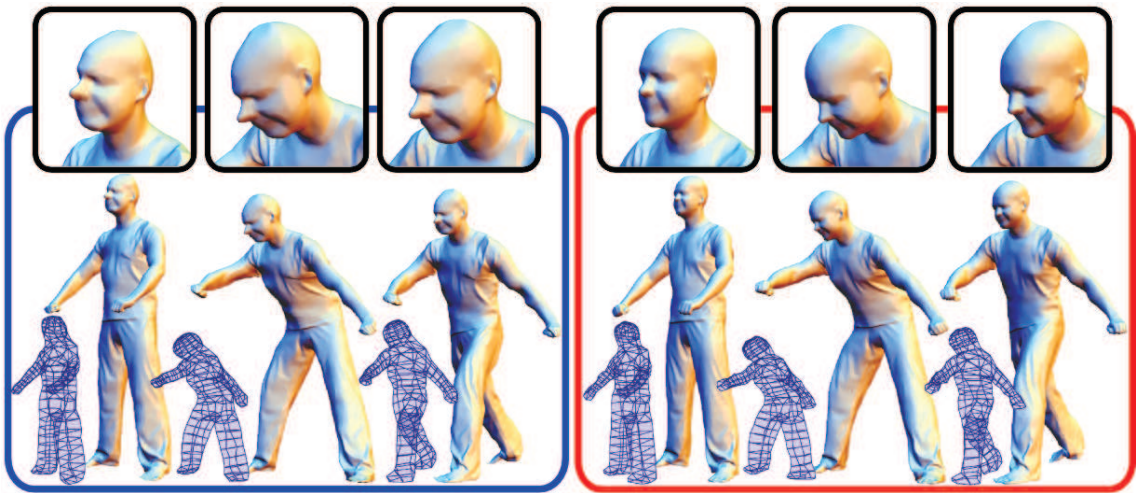


Figure 8.19: Cage and model reconstruction comparison between a globally regularized over-determined least-squares (blue) and our approach (red), with the same setting (MVC, Laplacian minimization, $\lambda = 1$).

spectral scheme enables localized regularization, but it also automatically focuses on the cage vertices where higher instabilities occur.

Fig. 8.16 shows the relative importance of each step of our approach (*MaxVol* and *spectral regularization*) on the *Skirt* sequence. The regularization parameters have been increased (s = 50) for illustration purpose. The full least-squares approach produces the largest spikes (1^{st} row). These are stabilized thanks to the MaxVol relaxation (2^{nd} row). Spectral regularization further improves the cage stability (3^{rd} and 4^{th} row), while the most stable cages are obtained in conjunction with MaxVol relaxation (4^{th} row).

For the *Capoeira* and *Horse* sequences (Fig. 8.14), our approach respectively yields an average L_2 error of 0.12 10^{-2} and 0.19 10^{-2} against 0.47 10^{-2} and 0.56 10^{-2} in the case of the skeleton-based framework by deAguiar et al. [23] (numbers from the original paper), which is more than a 50% error improvement. Although they are not directly comparable (different computers), the timings of our implementation are up to 50 times smaller than in [23], both on a pre-processing and per-frame basis.

Limitations As one can expect from a cage-based framework (error curves Fig. 8.14)), the maximum reconstruction error occurs on frames where the model pose is radically different (highly distorted) from the reference pose (e.g., 2nd row, middle). Another limitation is that, since our approach is cage-based, very small shape variations (e.g. small clothes motion) might be difficultly captured by the cage, since it tends to be designed for more global shape interactions.

Chapter 9

High-level representation for interactive modeling and processing of Meshes

It is common to use reduced representations for complex data analysis and processing. This is motivated by the fact that some problems have sometimes non-polynomial complexity, or have a complexity of high order, and are not directly applicable on full data.

We show some examples of applications of our high-level animated representation, and motivate the choice of cage representations in this context.

9.1 Animation lossy compression

The reconstruction step of our algorithm only requires $\mathcal{P}_R$, $\mathcal{C}_R$ and the output cage positions across time. For instance, the first animation of Fig. 8.14 can be encoded in binary format with 115 MB (single connectivity, 499 embeddings corresponding to the 499 frames of the animation), while it can be compressed down to 2.58 MB with our approach with a single connectivity, a single model embedding, a single cage connectivity and its 499 embeddings (see Fig. 9.1). The reconstruction error can be controlled with the spectrum threshold. Since our output cages smoothly vary over time, their vertex trajectories could further be compressed using orthogonal schemes (e.g. wavelets) for even higher compression rates. Traditional compression approaches, such as Progressive Meshes [40], require a clean connectivity on the animated mesh, while our approach can handle various shape representations (non-manifold meshes or point-sets for instance).

Discussion

Although different representations can be used, and obtain sometimes better compression factors, we found interesting to use cage coordinates to model the motion of a mesh.

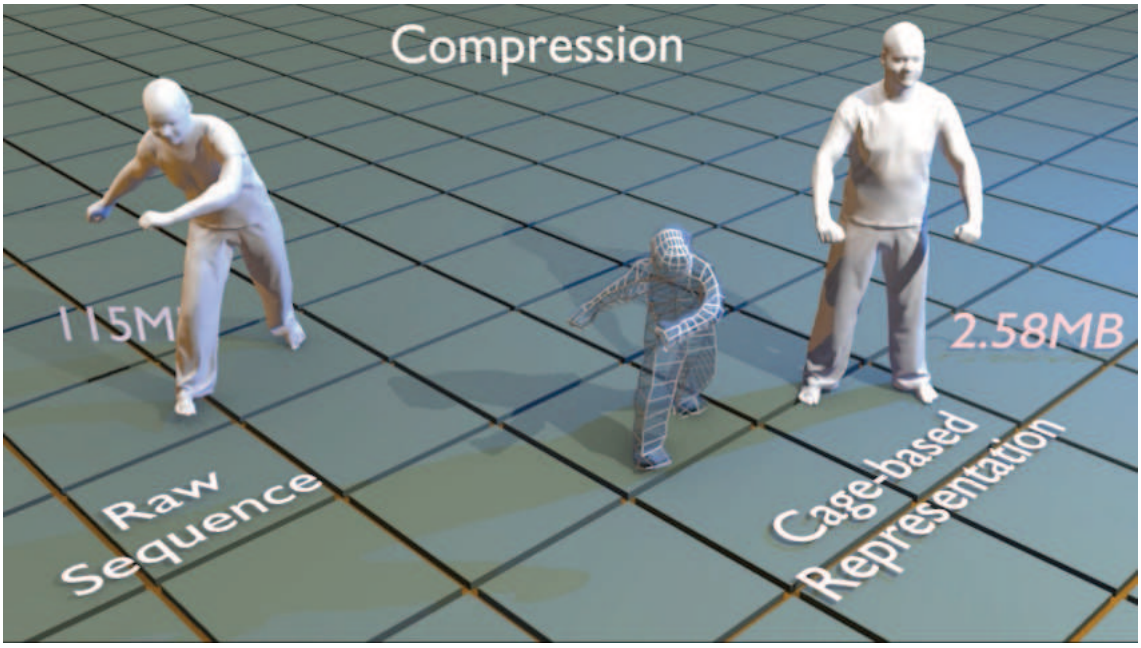


Figure 9.1: Compression of an animated mesh (499 frames). We need to store on disk one input mesh (connectivity + embedding), one connectivity for the cage and 499 cage embeddings. The compression factor for **storing** increases with an increasing number of frames. Note, that it has a upper bound which is the number of mesh vertices divided by the number of cage vertices.

Indeed, cage coordinates have been studied intensively over the past years with one main goal: **to produce visually pleasing deformations.**

This does not translate easily into mathematical terms, but it involves general mathematical notions such as harmonicity, biharmonicity, quasi-conformality, high order regularity and smoothness, $\cdots$ But in the end, artists make a choice between these deformation tools based on the visual quality they obtain.

Finally, recall that cage coordinate systems have been designed for producing animations of objects, and are used in the industry for the production of movies (e. g. "Ratatouille").

Bounds on the compression factor We note n_V (resp. n_T) the number of mesh vertices (resp. triangles) of the input shape, F the number of frames in the animation, and c_V (resp. c_T) the number of cage vertices (resp. triangles) that we use for the representation of the animated mesh. The complete animation can be stored by keeping **one frame of the input mesh** ($n_V + n_T$), and F times the number of cage vertices ($Fc_V + c_T$). The compression factor for **storing** the animation is then $\mathbf{Comp_{Store}} = \frac{n_T + Fn_V}{n_T + n_V + c_T + Fc_V}$.

- When the number of frames is high ($Fc_V >> n_V$), $\mathbf{Comp_{Store}} \simeq \frac{n_V}{c_V}$.
- When the number of frames is small ($Fc_V << n_V$), $\mathbf{Comp_{Store}} \simeq F$.

Additionally, we need to compute the cage coordinates matrix (size $n_V c_V$) when running the application. The compression factor when running an application is then $\mathbf{Comp_{Run}} = \frac{n_T + F n_V}{n_T + n_V c_V + F c_V}$.

- When the number of frames is high ($F >> \frac{n_V c_V}{n_V - c_V}$), $\mathbf{Comp_{Run}} \simeq \frac{n_V}{c_V}$.

To obtain a reduced representation **at run-time**, *the number of frames F needs to be higher than the lower bound* $\frac{n_V c_V}{n_V - c_V}$.

9.2 Animation transfer

Since they are clean and re-usable, the output cages can be employed for post-editing or animation transfer. An artist fits a model with complex topology (many handles) into the Capoeira reference cage. The cage sequence reconstructed with our algorithm automatically yields a smooth animation transfer to the new model.

Discussion

Cage-based deformation transfer has already been proposed by Ben-chen et al. [7] **in the context of rigid deformations**. Their method is based on an implicit framework (VHM [8]) that uses Green Coordinates as the underlying deformation machinery. The system optimizes the cage geometry in order to fit positional constraints on a subset of the vertices of the input shape, and optimizes for **unknown rotations on the medial axis**, as well as for **minimal Hessian**. Once they obtain the data of the Jacobian of the deformation on the medial axis of the *source shape*, by mapping manually the medial axis of the source shape and the target shape, they can transfer the Jacobian on the medial axis of the *target shape*. The same framework can be used to reconstruct the geometry of the target shape, thus transferring the animation.

The benefits of this approach are, that both objects can have a different cage embedding, and therefore they can handle more different geometries than our approach.

The main drawback is, that they are limited to **rigid animations**.

To represent animations obtained from performance capture, we experienced that Green Coordinates are not the right choice of cage coordinates, and Mean Value Coordinates and Harmonic Coordinates are better suited for such deformations. For these set of coordinates and the kind of motion we targetted, it seems difficult to rely only on the data of the Jacobian of the deformation on the medial axis.

9.3 Speeding up time-space processing tasks

Time-consuming processing tasks on time-varying 3D shapes can be drastically sped up by considering the cages reconstructed with our algorithm, which have a memory footprint orders of magnitude smaller than the input sequences. We present an application to the

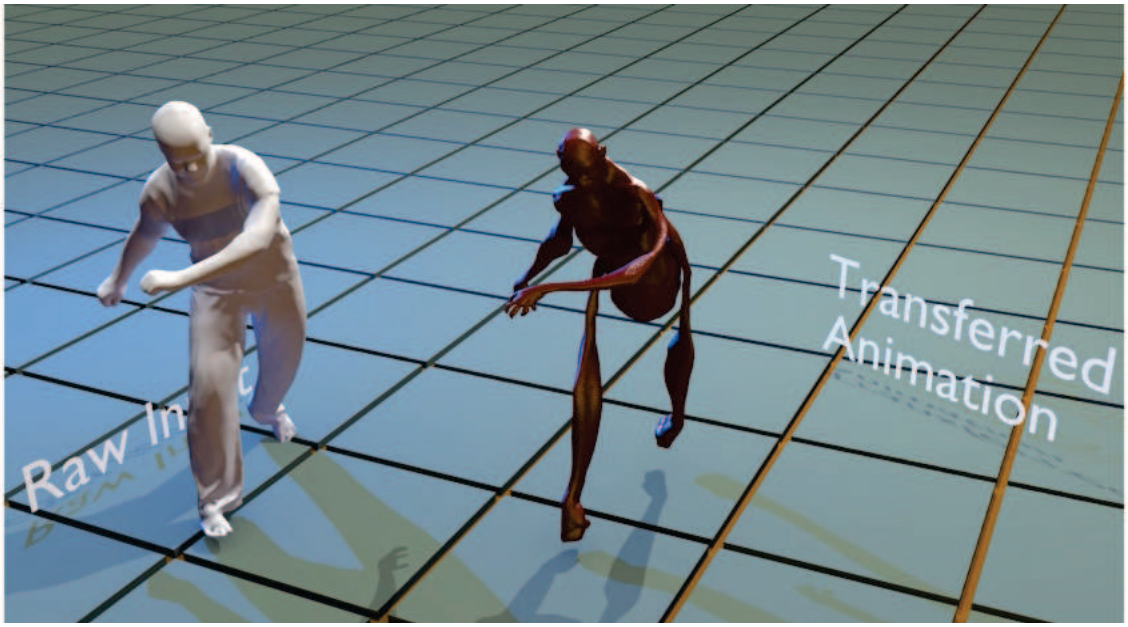


Figure 9.2: The embedding cage is a space that is slightly larger than the original mesh. By placing any object in that space, our cage representation allows to transfer in a straightforward manner animations from one mesh to another. Note, that this method does not work for highly dissimilar objects, as we require that both objects fit into the same cage for embedding (but they can be of different form, the target shape could be composed of thousands of stones).

interactive exploration of shape spaces [98] (implementation provided by the authors). Given a set of model poses and a cage, our algorithm automatically adapts the cage to the entire set of poses. Then, shape space exploration can be dones interactively (500x speed-up) on the space of cages, delegating detail preservation to the underlying coordinate system.

Discussion

The interpolation of many different meshes is an application that recent geometry processing techniques allows. Interpolating positions in a linear fashion is straightforward: it corresponds to create the segment between two points. Interpolating angles in two dimensions is already more difficult: they are defined up to 2π, and an infinity of solutions is possible (usually, the shortest path is considered though). Interpolating different poses of the same object is a highly challenging task, and the first (the main) difficulty is to give an appropriate definition to this problem.

Several solutions have been proposed with success [52, 98], but these require a significant amount of computations. We rely on the technique presented by Winkler et al. [98] (implementation gently provided by the authors), that interpolates linearly the lengths and the dihedral angles of the edges of the mesh, at different scales. This way, *differentials*

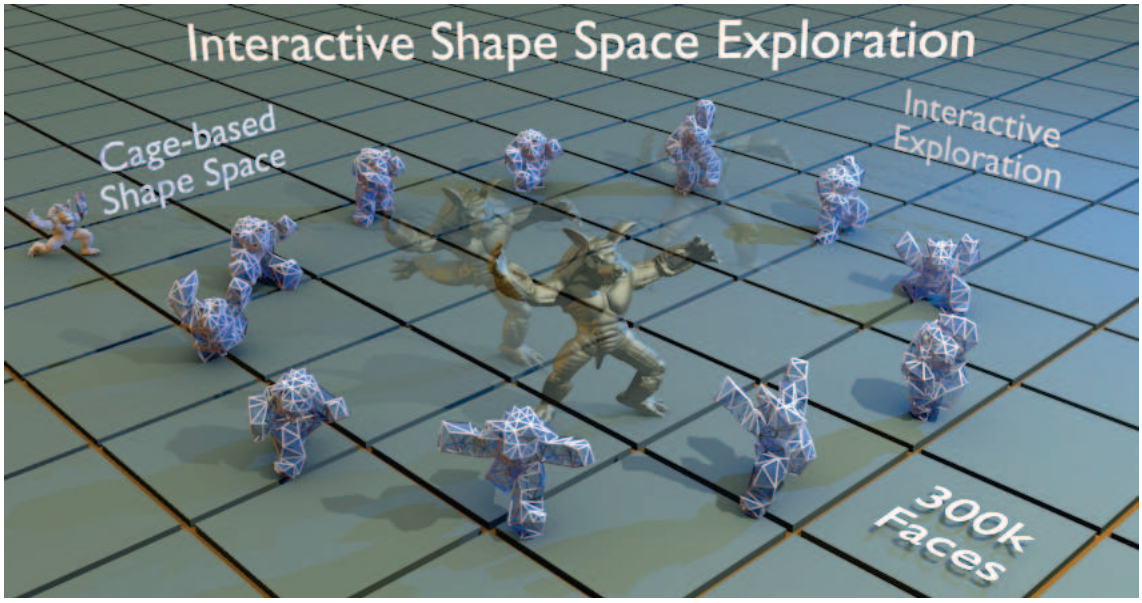


Figure 9.3: Real-time shape space exploration is performed on the set of cages that represent the different poses. The model is reconstructed on-the-fly on the GPU, as it requires only a multiplication of the cage coordinate matrix with the positions of the cage vertices that are obtained. We reach 20 frames per second on a standard laptop, allowing for the exploration of thousands of shapes within minutes.

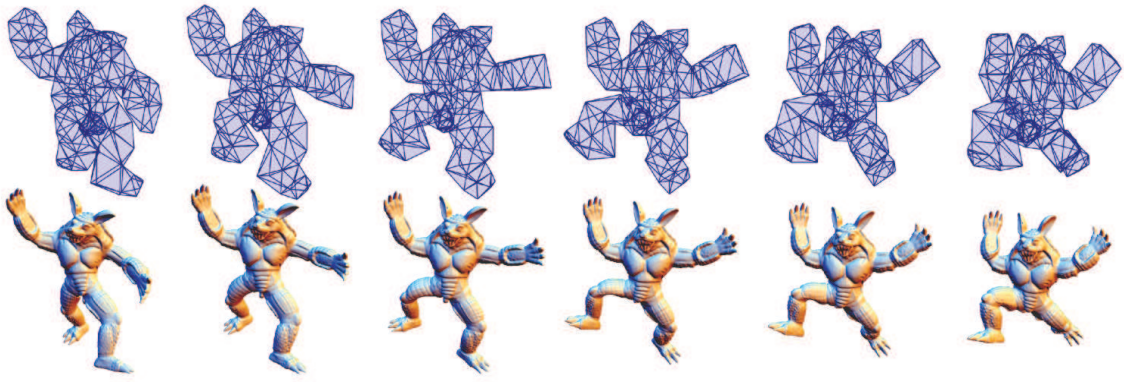


Figure 9.4: Results of shape space exploration performed on the cages and the reconstructed shapes. They can be used as the final result, or act only as a real-time feedback for the user to find the desired weights for the interpolation.

are actually blend together; it defines the manifold in an implicit way, therefore it involves quite some computations to retrieve the geometry from the interpolation.

The only "problem" is, that interpolating different meshes is **not intuitive**. We can actually say that, in this scenario, the user needs to explore the shape space that is defined. Practically, such an exploration is possible only if it is done **in real-time** and the user has an **immediate feedback**.

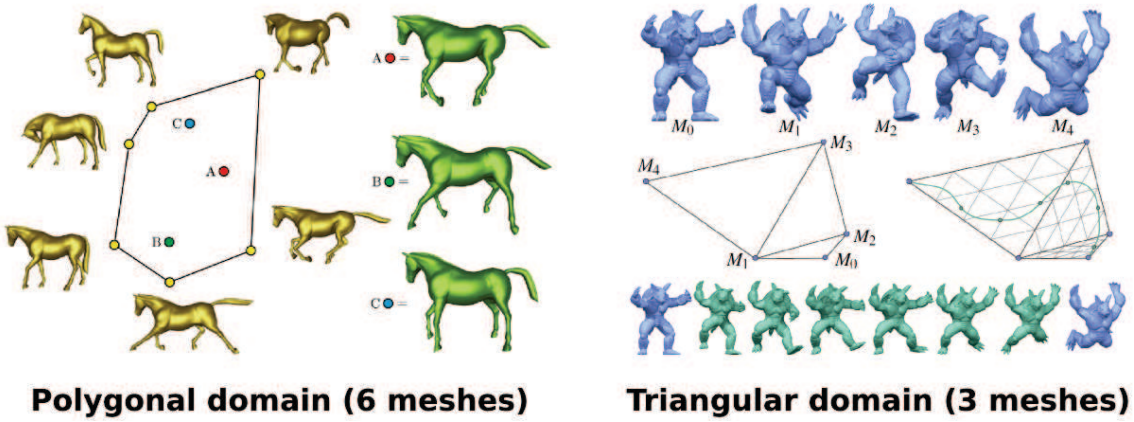


Figure 9.5: Left: Interpolation of 6 different meshes at the same time. Right: Interpolation of 3 different meshes at the same time. *Images taken from [98] and [52].*

Several scenarios that were presented in the previous work [52,98] show the interpolation of three different models at the same time (see Fig. 9.5), and none show the interpolation over 10 or 15 different meshes. We believe that it is for two main reasons: (i) the interpolation of many meshes requires generally a structure that is heavy and **memory consuming**, (ii) the interpolation of many meshes is very counter-intuitive.

These two remarks motivate the fact that **the space that defines the interpolation itself** (that gives a set of weights for the different meshes) should be editable as well, and in real-time.

In the example we presented, our cage representation is light enough to allow the interpolation of 11 different cages in real-time, and reconstruct the shape from the cage (all of this at 20 frames per second on a standard laptop). We use Mean Value Coordinates in two dimensions for the interactive definition of the weights, but we can rely on a 2D triangulation of the poses in the plane, such that each triangle t with vertices $\{t_0, t_1, t_2\}$ defines a domain for the linear blending of the three meshes $\{t_0, t_1, t_2\}$. As we precomputed the structure for the interpolation of the whole set of meshes, it corresponds simply to set the other weights to 0 in the interpolation. Using this machinery, the 2D space for the interpolation can be updated in real-time as well.

Note, that the reconstruction of the whole mesh relies on the cage coordinates system. In our application, we believe that the result is pleasing enough, and the images we provide were obtained using this strategy. But to allow such an exploration, recall that only **the exploration of the shape space needs to be performed in real-time** (the user explores sometimes thousands of different poses in our case). Once the weights are found, the user may want to use the original meshes instead of our cage representation, to obtain better results as a post-process.

9.4 Conclusion

We have presented CageR, an automatic algorithm for the compact and stable encoding of animated 3D shapes into cage-based representations. The main contributions are an optimal selection of handles for cage coordinate inversion and a novel spectral regularization scheme, which localizes its *destructive* effects and automatically focuses on the most unstable cage vertices. Our technique is fast (GPU implementation) and generates smoothly varying, re-usable cages which reconstruct the input sequence with comparable accuracy with previous methods. We demonstrated the benefits of our algorithm for several 3D+t processing tasks.

Interestingly, our approach makes only few geometrical considerations. Our solution naturally rises from a thorough algebraic analysis of the inverse problem. Hence, our framework completely delegates the geometrical aspects to the chosen cage coordinate model, guaranteeing the generality and the versatility of the technique.

Part V

Spatial coordinates analysis

In part IV, we considered cage coordinate systems as the underlying machinery for shape deformation. In this part, we present mathematical results on two sets of cage coordinates: (i) mean value coordinates, for which we present in chapter 10 an analytical solution for their derivatives and applications benefitting from them, and (ii) biharmonic coordinates, for which we present in chapter 11 an analytical form, following a construction similar to what has been done for Green Coordinates in the past.

This last part is rather technical compared to the three previous parts. Even though it presents applications in Computer Graphics, a large part of it is a technical derivation of derivatives and antiderivatives.

Chapter 10

Mean value coordinates derivatives

In this chapter, we present the mathematical derivation of the derivatives of the mean value coordinates, in 2D and in 3D. These can be used in any application using mean value coordinates, and goes beyond cage-based deformations.

We present some applications of these, but we focus more on the deformations, as it is an important part of our work. In chapter 8, we have used intensively this set of coordinates to represent animated 3D shapes using cages, as they are easy to use, since they do not require the shape to be entirely enclosed in the cage, and we did not studied fully the optimal creation of a cage embedding a shape. We then presented strategies that allow to use simple regularization terms on the cage (e. g. a null Laplacian everywhere on the cage's geometry) in this context.

The contributions made in this chapter can be used for CageR, by replacing regularization on the cage's geometry with a regularization term of the deformed space instead, by constraining some locations to have a low Hessian for example (which is also a regularization term that can be set in a linear system, and that is rotation-invariant).

10.1 Boundary value interpolation

Boundary value interpolation is a common problem in computer-aided design, simulation, visualization, computer graphics and geometry processing. Given a polygonal domain with prescribed scalar values on its boundary vertices, barycentric coordinate schemes enable computing a smooth interpolation of the boundary values at any point of the Euclidean space located in the interior (and possibly the exterior) of the domain. Such interpolations are efficiently obtained through a linear combination involving a weight (or *coordinate*) for each of the boundary vertices. These weights can be obtained by solving a system of linear equations [46] or, more efficiently, through a closed-form expression [34, 47].

However, in several applications, it may be desirable to enforce additional constraints on the interpolation, in particular constraints involving the partial derivatives of the interpolated function. Derivative constraints have been shown to provide additional flexibility to the interpolation problem and many optimization tasks can benefit from them. For instance,

Figure 10.1: "Kicking Demon". Red points indicate positional constraints; blue points indicate unknown rotational constraints. This pose was obtained by specifying only 14 positional constraints.
The derivation of the Jacobian and Hessian of the MVC coordinates makes it possible to induce *variational* MVC deformations (implicit cage deformation based on sparse user constraints, with rotation and smoothness enforcement).

in the context of *approximation* schemes based on Green Coordinates [61], constraints on the Jacobian and the Hessian have been used for implicit cage-based deformations [8]. In such a setting, given some sparse user constraints, an optimization process automatically retrieves an embedding of the polygonal domain (the *cage*) such that the transformation of the interior space is locally close to a rotation.

In order to achieve acceptable precision and time efficiency, such an optimization process requires a closed-form expression of the Jacobian and the Hessian of the interpolated function. While such closed-form expressions are known for *approximation* schemes (in particular for Green Coordinates [8, 95]), this is not the case for *interpolation* schemes. Moreover, these formulations are specific to the context of space deformation and it is not clear how to extend them to arbitrary functions.

We bridge this gap by deriving the closed-form expressions of the Jacobians and the Hessians of functions interpolated with Mean Value Coordinates (MVC) [34, 47], both for the 2D and 3D case. We also provide a complete analysis of their degenerate configurations along with accurate approximations of the derivatives for these configurations. We show the accuracy of this derivation with extensive numerical experiments.

We demonstrate the utility of this derivation for several applications, including cage-based implicit 3D model deformations (i.e. *Variational MVC deformations*). This technique allows for easy and interactive model deformations with sparse positional, rotational and smoothness constraints.

The cages produced by our algorithm can be directly re-used for further manipulations, which makes our framework directly compatible with existing software supporting MVC-based deformations.

10.1.1 Contributions

In this chapter, we present the following contributions:

1. the closed-form expressions of the Jacobian and the Hessian of Mean Value Coordinates, both in 2D and 3D – these expressions are both faster and more accurate than a variety of experimented Finite Difference schemes and they are not prone to numerical instability;

2. a thorough analysis of the degenerate configurations of these expressions, along with accurate alternate approximations for these configurations;

3. an implicit cage-based transformation technique using Mean Value Coordinates, called *Variational MVC Deformation*, which interactively optimizes the target cage embedding given sparse user positional constraints, while respecting smoothness and rotational constraints;

10.1.2 Overview

We first review Mean Value Coordinates in section 10.2. The core contribution of our work, the derivation of the Jacobian and Hessian, is presented in section 10.3 through 10.6. In particular, section 10.4 and 10.5 provide the specific results for the 2D and 3D cases respectively. Note, that we are mainly interested by the 3D case in this thesis, but the 2D case is much less involved than the 3D case, and it helps understanding the derivation for the 3D case, as the same particular cases need to be considered.

As the derivation of the Jacobian and the Hessian is relatively involved, for the reader's convenience, we highlighted the final expressions with rectangular boxes, whereas the final expressions for degenerate cases are highlighted with a ellipsoidal box. Note, that some derivation details were removed from the text for clarity purpose. These are given in Appendix at the end of the thesis.

Experimental evidence of the accuracy of our derivation is presented in section 10.7.

Finally we present applications demonstrating the utility of our contributions in section 10.8.

10.2 Background

In this section, we review the formulation of Mean Value Coordinates in 2D and 3D [47]. The reader that is not interested in a detailed presentation of this set of coordinates is referred to section 3.3.2 for a concise introduction of them. The following section will be useful to anyone who wants to follow in detail the mathematical derivation that we present in this chapter.

10.2.1 3D Mean Value Coordinates

Let η be a point in $\mathbb{R}^3$, and f a function expressed as a linear combination of some values $f_i \in \mathbb{R}^3$ defined at the vertices of a triangular mesh $\mathbf{M}$ by:

$$f(\eta) = \frac{\sum_i w_i(\eta) \cdot f_i}{\sum_i w_i(\eta)} = \sum_i \lambda_i(\eta) \cdot f_i$$

where $\lambda_i(\eta)$ is the *barycentric coordinate* of η with respect to the vertex i. It defines an interpolating function of the samples f_i if and only if $f(p_i) = f_i \forall i$.

For a point $x \in \mathbf{M}$ (a two-dimensional parameter), let $\phi_i[x]$ be the linear function on $\mathbf{M}$ which maps the vertex i to 1 and all other vertices to 0, and $p[x] \in \mathbb{R}^3$ the position of x on $\mathbf{M}$, and n_x its unit outward normal.

The definition of the coordinates λ_i should guarantee *linear precision* (i.e. $\eta = \sum_i \lambda_i(\eta) p_i$).

Since $\int_{B_\eta(\mathbf{M})} \frac{p[x]-\eta}{|p[x]-\eta|} dS_\eta(x) = 0$ (the integral of the unit outward normal onto the unit sphere is 0), the following equation holds:

$$\eta = \frac{\int_{B_\eta(\mathbf{M})} \frac{p[x]}{|p[x]-\eta|} dS_\eta(x)}{\int_{B_\eta(\mathbf{M})} \frac{1}{|p[x]-\eta|} dS_\eta(x)}$$

where $B_\eta(\mathbf{M})$ is the projection of $\mathbf{M}$ onto the unit sphere centered around η, and $dS_\eta(x)$ is the infinitesimal element of surface on this sphere at the projected point ($dS_\eta(x) = \frac{(p[x]-\eta)^t \cdot n_x}{||p[x]-\eta||^2} \cdot dx$).

By writing $p[x] = \sum_i \phi_i[x] p_i \ \forall x$, with $\sum_i \phi_i[x] = 1$, we have:

$$\eta = \frac{\sum_i \int_{B_\eta(\mathbf{M})} \frac{\phi_i[x]}{|p[x]-\eta|} dS_\eta(x) p_i}{\int_{B_\eta(\mathbf{M})} \frac{1}{|p[x]-\eta|} dS_\eta(x)}$$

The coordinates λ_i are then given by:

$$\lambda_i = \frac{\int_{B_\eta(\mathbf{M})} \frac{\phi_i[x]}{|p[x]-\eta|} dS_\eta(x)}{\int_{B_\eta(\mathbf{M})} \frac{1}{|p[x]-\eta|} dS_\eta(x)}$$

and the weights w_i such that $\lambda_i = \frac{w_i}{\sum_i w_i}$ are given by:

$$w_i = \int_{B_\eta(\mathbf{M})} \frac{\phi_i[x]}{|p[x] - \eta|} dS_\eta(x) \tag{10.1}$$

This definition guarantees linear precision [47]. It also provides a linear interpolation of the function prescribed at the vertices of the cage onto its triangles and it smoothly extends it to the entire 3D space.

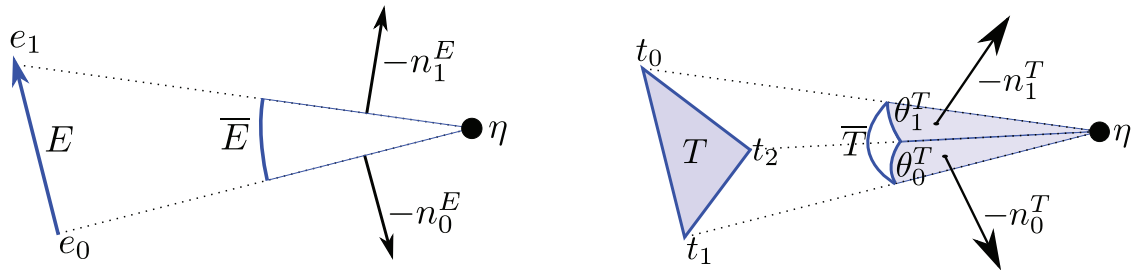


Figure 10.2: Spherical edge $\bar{\mathrm{E}}$ (left) and triangle $\bar{\mathrm{T}}$(right).

Weight computation

The support of the function $\phi_i[x]$ is only composed of the adjacent triangles to the vertex i. Eq. 10.1 can be re-written as $w_i = \sum_{T \in N1(i)} w_i^T$, with

$$w_i^T = \int_{B_\eta(T)} \frac{\phi_i[x]}{|p[x] - \eta|} \cdot d\overline{T} \tag{10.2}$$

Given a triangle T with vertices t_1, t_2, t_3, the following equation holds:

$$\begin{aligned} \sum_j w_{t_j}^T \cdot (p_{t_j} - \eta) &= \int_{B_\eta(T)} \frac{\sum_j \phi_{t_j}[x] \cdot (p_{t_j} - \eta)}{|p[x] - \eta|} \cdot d\overline{T} \\ &= \int_{B_\eta(T)} \frac{p[x] - \eta}{|p[x] - \eta|} \cdot d\overline{T} \triangleq m^T \end{aligned} \tag{10.3}$$

The latter integral is the integral of the unit outward normal on the spherical triangle $\overline{T}$. By noting the unit normal as $n_i^T = \frac{N_i^T}{|N_i^T|}$, with $N_i^T \triangleq (p_{t_{i+1}} - \eta) \wedge (p_{t_{i+2}} - \eta)$ (see Fig.10.2), m^T is given by (the integral of the unit normal on a closed surface is always 0):

$$m^T = \sum_i \frac{1}{2} \theta_i^T n_i^T \tag{10.4}$$

As suggested in [47], the weights $w_{t_j}^T$ can be obtained by noting A^T the 3x3 matrix $\{p_{t_1} - \eta, p_{t_2} - \eta, p_{t_3} - \eta\}$ (where ^{t} denotes the transpose):

$$\{w_{t_1}^T, w_{t_2}^T, w_{t_3}^T\}^t = A^{T^{-1}} \cdot m^T$$

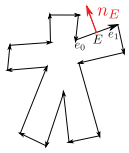
Since $N_i^{T^t} \cdot (p_{t_j} - \eta) = 0 \ \ \forall i \neq j$ and $N_i^{T^t} \cdot (p_{t_i} - \eta) = \det(A^T) \ \ \forall i$, the final expression for the weights is given by:

$$w_{t_i}^T = \frac{N_i^{T^t} \cdot m^T}{N_i^{T^t} \cdot (p_{t_i} - \eta)} = \frac{N_i^{T^t} \cdot m^T}{\det(A^T)} \quad \forall \eta \notin Support(T)$$

10.2.2 2D Mean Value Coordinates

In 2D, the orientation of an edge $E = e_0e_1$ of a closed polygon is defined by the normal n_E:

$$n_E = \frac{R_{\frac{\pi}{2}}(p_{e_1} - p_{e_0})}{|p_{e_1} - p_{e_0}|}$$



It defines consistently the *interior* and the *exterior* of the closed polygon. Then, similarly to the 3D case:

$$\sum_j w_{e_j}^E \cdot (p_{e_j} - \eta) = m^E = \sum_j n_j^E \tag{10.5}$$

with:

$$n_j^E = \frac{N_j^E}{|N_j^E|},\ N_0^E = R_{\frac{\pi}{2}}(\eta - p_{e_0}),\ N_1^E = -R_{\frac{\pi}{2}}(\eta - p_{e_1})$$

Therefore:

$$m^E = R_{\frac{\pi}{2}}\left(\frac{\eta - p_{e_0}}{|\eta - p_{e_0}|} - \frac{\eta - p_{e_1}}{|\eta - p_{e_1}|}\right) \tag{10.6}$$

Since $(p_{e_j} - \eta)^t \cdot N_j^E = 0$ (Fig.10.2), we obtain w_i^E with:

$$w_{e_i}^E = \frac{m^{E^t} \cdot N_{i+1}^E}{(p_{e_i} - \eta)^t \cdot N_{i+1}^E}$$

10.3 Derivation Overview

In the following, we present our main contribution: the derivation of the Jacobians and the Hessians of Mean Value Coordinates. In this section, we briefly give an overview of the derivation.

Let $f : \mathbf{M} \to \mathbb{R}^n$ be a piecewise linear field defined on $\mathbf{M}$ (in 2D, $\mathbf{M}$ is a closed polygon, in 3D, $\mathbf{M}$ is a closed triangular mesh). As reviewed in the previous section, f can be smoothly interpolated with Mean Value Coordinates for any point η of the Euclidean space:

$$f(\eta) = \sum_i \lambda_i \cdot f(p_i)$$

Then, the Jacobian and the Hessian of f, respectively noted Jf and Hf, are expressed as the linear tensor product of the values $f(p_i)$ with the gradient $\overrightarrow{\nabla}\lambda_i$ and the Hessian $H\lambda_i$ of the coordinates respectively:

$$\begin{cases} Jf & = \sum_i f(p_i) \cdot \overrightarrow{\nabla}\lambda_i{}^t \\ Hf & = \sum_i f(p_i) \cdot H\lambda_i \end{cases}$$

Note, that care must be taken when writing $Hf = \sum_i f(p_i) \cdot H\lambda_i$, as it is correct for a scalar function, but it is ambiguous for a multi-dimensional function. To be correct, the Hessian of each k^{th} coordinate f_k of f is given by $Hf_k(\eta) = \sum_i f_k(p_i) \cdot H\lambda_i(\eta)$.

Since $\lambda_i = \frac{w_i}{\sum_j w_j}$,

$$\overrightarrow{\bigtriangledown}\lambda_i = \frac{\overrightarrow{\bigtriangledown} w_i}{\sum_j w_j} - \frac{w_i \cdot \sum_j \overrightarrow{\bigtriangledown} w_j}{(\sum_j w_j)^2} \tag{10.7}$$

Then, the Hessian can be obtained with the following equations

$$\begin{aligned} H\lambda_i = & \frac{Hw_i}{\sum_j w_j} - \frac{w_i \sum_j Hw_j}{(\sum_j w_j)^2} \\ & - \frac{\overrightarrow{\bigtriangledown} w_i \cdot \sum_j \overrightarrow{\bigtriangledown}(w_j)^t + \sum_j \overrightarrow{\bigtriangledown}(w_j) \cdot \overrightarrow{\bigtriangledown} {w_i}^t}{(\sum_j w_j)^2} \\ & + \frac{2w_i(\sum_j \overrightarrow{\bigtriangledown} w_j) \cdot (\sum_j \overrightarrow{\bigtriangledown} w_j)^t}{(\sum_j w_j)^3} \end{aligned} \tag{10.8}$$

The above expressions are general and valid for the 2D and 3D cases. Thus, in order to derive a closed-form expression of the gradient and the Hessian of the Mean Value Coordinates λ_i, one needs to derive the expressions of $\overrightarrow{\bigtriangledown} w_i$ (Eq. 10.7) and Hw_i (Eq. 10.8). The expressions of these terms are derived in section 10.4.1 and section 10.4.2 respectively for the 2D case and in section 10.5.1 and section 10.5.2 for the 3D case.

Properties

Functions interpolated by means of Mean Value Coordinates as previously described have the following properties:

1. they are interpolant on $\mathbf{M}$
2. they are defined everywhere in $\mathbb{R}^n$
3. they are C^∞ everywhere not on $\mathbf{M}$
4. they are C^0 on the edges (resp. vertices) of $\mathbf{M}$ in 3D (resp. in 2D)

Since these are interpolant of piecewise linear functions defined on a piecewise linear domain, they cannot be differentiable on the edges of the triangles (resp. the vertices of the edges) of the cage in 3D (resp. in 2D). Although, as they are continuous everywhere, they may admit in these cases **directional derivatives** like for almost all continuous functions, but as they are of no use at all in general, we won't present these quantities. Recall that the directional derivative is the value $\partial f_u(\eta) = \lim_{\epsilon \to 0^+} \frac{f(\eta + \epsilon \cdot u) - f(\eta)}{\epsilon}$, with $u \in \mathbb{R}^3, ||u|| = 1, \epsilon \in \mathbb{R}$, **which strongly depends on the orientation of the vector** u

where the limit is considered. These derivatives cannot be used to evaluate the neighborhood of a the function around the point in general with a single gradient (or Jacobian if the function is multi-dimensional).

In the following, we provide formulae for the 1^{st} and 2^{nd} order derivatives of the Mean Value Coordinates everywhere in space but on the cage.

10.4 MV-Gradients and Hessians in 2D

10.4.1 Expression of the MV-gradients

By deriving Eq. 10.5, we obtain:

$$\sum_j (p_{e_j} - \eta) \cdot \overrightarrow{\nabla} w_{e_j}^{E\,t} = Jm^E + \sum_j w_{e_j}^E I_2 = B^E(\eta) \tag{10.9}$$

Jm^E is given by deriving Eq. 10.6:

$$Jm^E = R_{\frac{\pi}{2}}\left(\frac{I_2}{|\eta - p_{e_0}|} - \frac{I_2}{|\eta - p_{e_1}|} - \frac{(\eta - p_{e_0}) \cdot (\eta - p_{e_0})^t}{|\eta - p_{e_0}|^3} + \frac{(\eta - p_{e_1}) \cdot (\eta - p_{e_1})^t}{|\eta - p_{e_1}|^3}\right) \tag{10.10}$$

Then, in the general case where $(p_{e_i} - \eta)^t \cdot N_{i+1}^E \neq 0$, the gradient of the weights is given by the following expression:

$$\boxed{\overrightarrow{\nabla} w_{e_i}^E = \frac{B^{E\,t} \cdot N_{i+1}^E}{(p_{e_i} - \eta)^t \cdot N_{i+1}^E}} \tag{10.11}$$

Special case: $\eta \in (e_0e_1), \notin [e_0e_1]$

The special case where $(p_{e_i} - \eta)^t \cdot N_{i+1}^E = 0$ only occurs when η lies on the same line as the edge E (η lies on the *support* of the edge E, noted $Support(E)$). As discussed in section 10.3, we omit the case where $\eta \in [e_0e_1]$. Since the length of $\bar{\mathrm{E}}$ is zero (see Fig. 10.2), for all $\eta \in (e_0e_1), \notin [e_0e_1]$, $w_{e_i}^E(\eta) = 0\ \forall i = 0, 1$.

Similarly, $\overrightarrow{\nabla} w_{e_i}^E$ and n_E are collinear, then:

$$\overrightarrow{\nabla} w_{e_i}^E = \frac{\partial(w_{e_i}^E(\eta + \epsilon n_E))}{\partial \epsilon}_{|\epsilon \to 0} n_E$$

A closed-form expression can be derived from the above equation with Taylor expansions. For conciseness, the details of this derivation are given in Appendix C (see section 13.3) and only the final expression is given here:

$$\overrightarrow{\bigtriangledown} w_{e_i}^E = (\sum_j \frac{N_j^{E^t} \cdot N_{i+1}^E}{2|E||N_j^E|^3} + \frac{(-1)^{i+j}}{|E||N_j^E|}) n_E \qquad (10.12)$$
$$\forall \eta \in (e_0 e_1), \notin [e_0 e_1]$$

10.4.2 Expression of the MV-Hessians

By deriving Eq. 10.9 successively with regards to $c = \{x, y\}$, we obtain:

$$\begin{aligned} &\sum_i \partial_c (p_{e_i} - \eta) \cdot \overrightarrow{\bigtriangledown} w_{e_i}^{E^t} + \sum_i (p_{e_i} - \eta) \cdot \partial_c (\overrightarrow{\bigtriangledown} w_{e_i}^{E^t}) \\ &= \partial_c (Jm^E) + \sum_i \partial_c (w_{e_i}^E) \cdot I_2 \end{aligned}$$

$$\left\{ \begin{array}{ll} \sum_i (p_{e_i} - \eta) \cdot \partial_x (\overrightarrow{\bigtriangledown} w_{e_i}^{E^t}) & = C_x^E \\ \sum_i (p_{e_i} - \eta) \cdot \partial_y (\overrightarrow{\bigtriangledown} w_{e_i}^{E^t}) & = C_y^E \end{array} \right. \qquad (10.13)$$

with

$$\left\{ \begin{array}{ll} C_x^E & = \sum_i \begin{pmatrix} 1 \\ 0 \end{pmatrix} \cdot \overrightarrow{\bigtriangledown} w_{e_i}^{E^t} + \partial_x (Jm^E) + \sum_i \partial_x (w_{e_i}^E) \cdot I_2 \\ C_y^E & = \sum_i \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot \overrightarrow{\bigtriangledown} w_{e_i}^{E^t} + \partial_y (Jm^E) + \sum_i \partial_y (w_{e_i}^E) \cdot I_2 \end{array} \right.$$

Expressions have been provided for all of the terms appearing in the above equations, except for the derivatives of Jm^E.

Expression of $\partial_c(Jm^E)$

By deriving Eq. 10.10, we obtain:

$$\begin{aligned} \partial_c (Jm^E) = & R_{\frac{\pi}{2}} \cdot \Big(\frac{(\eta - p_{e_1})_{(c)} I_2}{|\eta - p_{e_1}|^3} - \frac{(\eta - p_{e_0})_{(c)} I_2}{|\eta - p_{e_0}|^3} - \frac{\delta_c \cdot (\eta - p_{e_0})^t + (\eta - p_{e_0}) \cdot {\delta_c}^t}{|\eta - p_{e_0}|^3} \\ & + \frac{3(\eta - p_{e_0})_{(c)} (\eta - p_{e_0}) \cdot (\eta - p_{e_0})^t}{|\eta - p_{e_0}|^5} + \frac{\delta_c \cdot (\eta - p_{e_1})^t + (\eta - p_{e_1}) \cdot {\delta_c}^t}{|\eta - p_{e_1}|^3} \\ & - \frac{3(\eta - p_{e_1})_{(c)} (\eta - p_{e_1}) \cdot (\eta - p_{e_1})^t}{|\eta - p_{e_1}|^5} \Big) \end{aligned}$$

with $\delta_x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\delta_y = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Final expression of $Hw_{e_i}^E$

From Eq. 10.13, ($(p_{e_i} - \eta)^t \cdot N_i^E = 0$, see Fig. 10.2) we obtain:

$$\left\{ \begin{array}{ll} \partial_x(\overrightarrow{\bigtriangledown} w_{e_i}^E) & = \frac{C_x^{E^t} \cdot N_{i+1}^E}{(p_{e_i} - \eta)^t \cdot N_{i+1}^E} \\ \partial_y(\overrightarrow{\bigtriangledown} w_{e_i}^E) & = \frac{C_y^{E^t} \cdot N_{i+1}^E}{(p_{e_i} - \eta)^t \cdot N_{i+1}^E} \end{array} \right\} \quad \forall \eta \notin (p_{e_0} p_{e_1}).$$

Finally:

$$\boxed{Hw_{e_i}^E = \begin{pmatrix} \partial_x(\overrightarrow{\bigtriangledown} w_{e_i}^E)^t \\ \partial_y(\overrightarrow{\bigtriangledown} w_{e_i}^E)^t \end{pmatrix} \qquad (10.14)}$$

Special case: $\eta \in (e_0 e_1), \notin [e_0 e_1]$

When η lies on the support of the edge E, $\overrightarrow{\bigtriangledown} w_{e_i}^E(\eta) = dw_{e_i}^E n_E$, where $dw_{e_i}^E = \frac{\partial(w_{e_i}^E(\eta + \epsilon n_E))}{\partial \epsilon}|_{\epsilon \to 0}$ is a scalar term obtained in the special case of section 10.4.1 (Appendix C, see section 13.3).

Therefore $\forall \eta \in (e_0 e_1), \notin [e_0 e_1]$:

$$Hw_{e_i}^E = \overrightarrow{\bigtriangledown}(dw_{e_i}^E) \cdot n_E^t$$

$$\begin{aligned} \overrightarrow{\bigtriangledown}(dw_{e_i}^E) = & \sum_j \frac{JN_{i+1}^{E\ t} \cdot N_j^E + JN_j^{E^t} \cdot N_{i+1}^E}{2|E||N_j^E|^3} \\ & - \sum_j \frac{3(N_j^{E^t} \cdot N_{i+1}^E) JN_j^{E^t} \cdot N_j^E}{2|E||N_j^E|^5} \\ & + \sum_j \frac{(-1)^{i+j+1} JN_j j^{E^t} \cdot N_j^E}{|E||N_j^E|^3} \end{aligned}$$

$$\begin{aligned} \overrightarrow{\bigtriangledown}(dw_{e_i}^E) = & R_{\frac{\pi}{2}} \cdot \sum_j \frac{3(-1)^{i+1} N_j^E + (-1)^j N_{i+1}^E}{2|E||N_j^E|^3} \\ & + R_{\frac{\pi}{2}} \cdot \sum_j \frac{3(-1)^{j+1} (N_j^{E^t} \cdot N_{i+1}^E) N_j^E}{2|E||N_j^E|^5} \end{aligned}$$

Finally:

$$\begin{aligned} Hw_{e_i}^E = R_{\frac{\pi}{2}} \cdot \sum_j \frac{3(-1)^{i+1} N_j^E + (-1)^j N_{i+1}^E}{2|E||N_j^E|^3} \cdot {n_E}^t \\ + R_{\frac{\pi}{2}} \cdot \sum_j \frac{3(-1)^{j+1} (N_j^{Et} \cdot N_{i+1}^E) N_j^E}{2|E||N_j^E|^5} \cdot {n_E}^t \\ \forall \eta \in (e_0 e_1), \notin [e_0 e_1] \end{aligned} \tag{10.15}$$

10.5 MV-Gradients and Hessians in 3D

10.5.1 Expression of the MV-Gradients

Instead of deriving Eq. 10.2 (which involves the calculus of an integral), we derive the Jacobian from Eq. 10.3 (by noting $B^T \triangleq Jm^T + \sum_j w_{t_j}^T \cdot I_3$):

$$\sum_j (p_{t_j} - \eta) \cdot \overrightarrow{\bigtriangledown} {w_{t_j}^T}^t = B^T \tag{10.16}$$

From the above expression, we obtain: (again, $N_i^{Tt} \cdot (p_{t_j} - \eta) = 0 \; \forall j \neq i$, see Fig. 10.2)

$$\boxed{\begin{aligned} \overrightarrow{\bigtriangledown} w_{t_i}^T = \frac{{B^T}^t \cdot N_i^T}{\det(A^T)} \\ \forall \eta \notin Support(T) \end{aligned}} \tag{10.17}$$

At this point, the Jacobian of the vector m^T is required for the expression of B^T.

Expression of Jm^T

From Eq. 10.4, we have:

$$m^T = \sum_j \frac{\theta_j^T n_j^T}{2} = \sum_j \frac{\theta_j^T N_j^T}{2|N_j^T|}$$

Then, we obtain:

$$\begin{aligned} Jm^T = & \sum_j \frac{N_j^T \cdot \overrightarrow{\bigtriangledown} {\theta_j^T}^t}{2|N_j^T|} + \sum_j \frac{\theta_j^T J N_j^T}{2|N_j^T|} \\ & - \sum_j \frac{\theta_j^T N_j^T \cdot (J {N_j^T}^t \cdot N_j^T)^t}{2|N_j^T|^3} \end{aligned} \tag{10.18}$$

Derivatives of N_j^T

From the expression of N_j^T (section 10.2.1), we obtain:

$$\begin{aligned} N_j^T(\eta + d\eta) =& (p_{t_{j+1}} - \eta - d\eta) \times (p_{t_{j+2}} - \eta - d\eta) \\ =& (p_{t_{j+1}} - \eta) \times (p_{t_{j+2}} - \eta) + (p_{t_{j+2}} - p_{t_{j+1}}) \times d\eta. \end{aligned}$$

Therefore

$$JN_j^T = (p_{t_{j+2}} - p_{t_{j+1}})_{[\wedge]}$$

where $k_{[\wedge]}$ is the skew 3×3 matrix (i.e. ${k_{[\wedge]}}^t = -k_{[\wedge]}$) such that $k_{[\wedge]} \cdot u = k \times u \ \ \forall k, u \in \mathbb{R}^3$.

In particular, we see from Eq. 10.19 that N_j^T *admits a null second order derivative.*

Expression of $\overrightarrow{\bigtriangledown}\theta_j^T$

The term $\overrightarrow{\bigtriangledown}\theta_j^T$ can be derived from the following expressions:

$$\begin{array}{lll} \sin(\theta_j^T) = S_j^T & , \quad S_j^T & \triangleq \frac{|(p_{t_{j+2}} - \eta) \times (p_{t_{j+1}} - \eta)|}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|} = \frac{|N_j^T|}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|} \\ \cos(\theta_j^T) = C_j^T & , \quad C_j^T & \triangleq \frac{(p_{t_{j+2}} - \eta)^t \cdot (p_{t_{j+1}} - \eta)}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|} \end{array}$$

$$\cos(\theta_j^T) \cdot \overrightarrow{\bigtriangledown}\theta_j^T = \overrightarrow{\bigtriangledown} S_j^T \tag{10.19}$$

$$-\sin(\theta_j^T) \cdot \overrightarrow{\bigtriangledown}\theta_j^T = \overrightarrow{\bigtriangledown} C_j^T \tag{10.20}$$

$\overrightarrow{\bigtriangledown}\theta_j^T$ can be evaluated with Eq. 10.19 when $\theta_j^T \neq \pi/2$, and with Eq. 10.20 when $\theta_j^T \neq 0, \pi$. $\overrightarrow{\bigtriangledown} S_j^T$ and $\overrightarrow{\bigtriangledown} C_j^T$ are given by the following expressions:

$$\overrightarrow{\bigtriangledown} S_j^T = \frac{JN_j^{T^t} \cdot N_j^T}{|N_j^T| \cdot |p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|} - \frac{|N_j^T| \cdot (\eta - p_{t_{j+2}})}{|p_{t_{j+2}} - \eta|^3 \cdot |p_{t_{j+1}} - \eta|} - \frac{|N_j^T| \cdot (\eta - p_{t_{j+1}})}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|^3}$$

$$\overrightarrow{\bigtriangledown} S_j^T = \frac{JN_j^{T^t} \cdot N_j^T}{|N_j^T| \cdot |p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|} - \frac{(\eta - p_{t_{j+2}}) \cdot \sin(\theta_j^T)}{|p_{t_{j+2}} - \eta|^2} - \frac{(\eta - p_{t_{j+1}}) \cdot \sin(\theta_j^T)}{|p_{t_{j+1}} - \eta|^2} \tag{10.21}$$

$$\begin{aligned} \overrightarrow{\bigtriangledown} C_j^T =& \frac{2\eta - p_{t_{j+1}} - p_{t_{j+2}}}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|} - \frac{(\eta - p_{t_{j+2}}) \cdot (p_{t_{j+2}} - \eta)^t \cdot (p_{t_{j+1}} - \eta)}{|p_{t_{j+2}} - \eta|^3 \cdot |p_{t_{j+1}} - \eta|} \\ & - \frac{(\eta - p_{t_{j+1}}) \cdot (p_{t_{j+2}} - \eta)^t \cdot (p_{t_{j+1}} - \eta)}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|^3} \end{aligned}$$

$$\overrightarrow{\bigtriangledown} C_j^T = \frac{2\eta - p_{t_{j+1}} - p_{t_{j+2}}}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|} - \frac{(\eta - p_{t_{j+2}}) \cdot \cos(\theta_j^T)}{|p_{t_{j+2}} - \eta|^2} - \frac{(\eta - p_{t_{j+1}}) \cdot \cos(\theta_j^T)}{|p_{t_{j+1}} - \eta|^2} \tag{10.22}$$

From Eq. 10.19 and 10.20, we obtain:

$$\cos(\theta_j^T)\overrightarrow{\bigtriangledown} S_j^T - \sin(\theta_j^T)\overrightarrow{\bigtriangledown} C_j^T = \cos(\theta_j^T)^2\overrightarrow{\bigtriangledown}\theta_j^T + \sin(\theta_j^T)^2\overrightarrow{\bigtriangledown}\theta_j^T = \overrightarrow{\bigtriangledown}\theta_j^T$$

And by replacing the expressions of $\overrightarrow{\bigtriangledown} S_j^T$ and $\overrightarrow{\bigtriangledown} C_j^T$ on the left side of the equality by those of Eq. 10.21 and 10.22:

$$\begin{aligned}\overrightarrow{\bigtriangledown}\theta_j^T &= \frac{\cos(\theta_j^T) J N_j^{T^t} \cdot N_j^T}{|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta||N_j^T|} - \frac{\sin(\theta_j^T)(2\eta - p_{t_{j+2}} - p_{t_{j+1}})}{|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|} \\ &= \frac{\cos(\theta_j^T)\sin(\theta_j^T) J N_j^{T^t} \cdot N_j^T}{|N_j^T|^2} - \frac{\sin(\theta_j^T)^2(2\eta - p_{t_{j+2}} - p_{t_{j+1}})}{|N_j^T|}\end{aligned}$$

$$\overrightarrow{\bigtriangledown}\theta_j^T = \frac{\cos(\theta_j^T)\sin(\theta_j^T) J N_j^{T^t} \cdot N_j^T}{|N_j^T|^2} - \frac{\sin(\theta_j^T)^2(2\eta - p_{t_{j+2}} - p_{t_{j+1}})}{|N_j^T|} \tag{10.23}$$

At this point, an expression for $\overrightarrow{\bigtriangledown}\theta_j^T$ has been provided. To complete the expression of Jm^T, Eq. 10.23 and 10.18 need to be combined:

$$\begin{aligned}Jm^T = &\sum_j \frac{\cos(\theta_j^T)\sin(\theta_j^T) N_j^T \cdot (J N_j^{T^t} \cdot N_j^T)^t}{2|N_j^T|^3} \\ &- \sum_j \frac{\sin(\theta_j^T)^2 N_j^T \cdot (2\eta - p_{t_{j+2}} - p_{t_{j+1}})^t}{2|N_j^T|^2} \\ &+ \sum_j \frac{\theta_j^T J N_j^T}{2|N_j^T|} - \sum_j \frac{\theta_j^T N_j^T \cdot (J N_j^{T^t} \cdot N_j^T)^t}{2|N_j^T|^3}\end{aligned}$$

Final expression of Jm^T

We obtain the final expression of $Jm^T(\eta)$ as:

$$\begin{aligned}Jm^T = &\sum_j \frac{e_1(\theta_j^T) N_j^T \cdot N_j^{T^t} \cdot J N_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} \\ &- \sum_j \frac{N_j^T \cdot (2\eta - p_{t_{j+1}} - p_{t_{j+2}})^t}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^2} \\ &+ \sum_j \frac{e_2(\theta_j^T) J N_j^T}{2|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|}\end{aligned}$$

where $e_1 = \frac{\cos(x)\sin(x)-x}{\sin(x)^3}$ and $e_2(x) = \frac{x}{\sin(x)}$ are two well-defined functions on $]0,\pi[$ which admit well-controlled Taylor expansions around 0.

Given the final expression of Jm^T (Eq. 10.24), we recall that $\overrightarrow{\bigtriangledown} w_{t_i}$ can be computed with the following expression: $\overrightarrow{\bigtriangledown} w_{t_i}^T = \frac{B^{T^t} \cdot N_i^T}{\det(A^T)}$ (with $B^T = Jm^T + \sum_j w_{t_j}^T \cdot I_3$).

Special case: $\eta \in Support(T), \notin T$

The expressions provided so far admit degenerate cases when $\det(A^T) = 0$ (Eq. 10.17). Similarly to the 2D setting, these cases only occur when η is lying on the support plane of T, noted $Support(T)$.

For $\eta \in Support(T), \notin T$, as discussed above, given small steps in the support of T, the weights are set to 0: $\forall (\eta + d\eta) \in Support(T), \notin T$, $w_{t_i}^T(\eta + d\eta) = 0$.

Therefore, the weights only evolve in the direction of the normal of T:

$$\forall \eta \in Support(T), \notin T: \quad \overrightarrow{\bigtriangledown} w_{t_i}^T = \frac{\partial w_{t_i}^T(\eta + \epsilon n_T)}{\partial \epsilon}_{|\epsilon \to 0} \cdot n_T$$

To approximate the above expression, we consider the Taylor expansion of $w_{t_i}^T(\eta + \epsilon n_T) = \frac{N_i^T(\eta+\epsilon n_T)^t \cdot m^T(\eta+\epsilon n_T)}{\det(A^T(\eta+\epsilon n_T))}$ with regards to ϵ.

The details of this expansion can be found in Appendix A (see section 13.1). The final result is given by:

$$\begin{aligned}\overrightarrow{\bigtriangledown} w_i^T = &- \sum_j \frac{e_2(\theta_j^T)(p_{t_{i+2}} - p_{t_{i+1}})^t \cdot (p_{t_{j+2}} - p_{t_{j+1}})}{4|T||p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|} n_T \\ &- \sum_j \frac{e_1(\theta_j^T)|p_{t_{j+2}} - p_{t_{j+1}}|^2 N_i^{T^t} \cdot N_j^T}{8|T|(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} n_T \\ &- \sum_j \frac{\cos(\theta_j^T) e_3(\theta_j^T) N_i^{T^t} \cdot N_j^T}{4|T|(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^2} n_T \\ &\forall \eta \in Support(T), \notin T\end{aligned}$$

with $e_2(x) = \frac{x}{\sin(x)}, e_1(x) = \frac{\cos(x)\sin(x)-x}{\sin(x)^3}$, and $e_3(x) = \frac{\cos(x)-1}{\sin(x)^2}$ being functions well defined on $]0,\pi[$ and which admit controllable Taylor expansion around 0.

10.5.2 Expression of the MV-Hessians

We note $\delta^x = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \delta^y = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \delta^z = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$.

Derivation of Hw_i^T

By deriving Eq. 10.16 successively by $c = x, y$, and z, we obtain:

$$\begin{aligned}&\sum_i \partial_c(p_{t_i} - \eta) \cdot \overrightarrow{\bigtriangledown} w_i^{T\,t} + \sum_i (p_{t_i} - \eta) \cdot \partial_c(\overrightarrow{\bigtriangledown} w_i^{T\,t}) \\ &= \partial_c(Jm^T) + \sum_i \partial_c(w_i^T) \cdot I_3\end{aligned}$$

$$\sum_j (p_{t_j} - \eta) \cdot \partial_c(\overrightarrow{\bigtriangledown} w_{t_j}^{T\,t}) = C_c^T \tag{10.24}$$

with $C_c^T \triangleq \delta^c \cdot \sum_j \overrightarrow{\bigtriangledown} w_{t_j}^{T\,t} + \partial_c(Jm^T) + \sum_j \partial_c(w_{t_j}^T) \cdot I_3$.

All these terms have been expressed previously, except for the derivatives of Jm^T.

Expression of the derivatives of Jm^T

Given a vector $k \in \mathbb{R}^3$, we note $(k)_{(x)}, (k)_{(y)}, (k)_{(z)}$ its components in x, y, and z.

By deriving Eq. 10.24 term by term, and using Eq. 10.23 for $\overrightarrow{\bigtriangledown} \theta_j^T$, we obtain the final expression of $\partial_c(Jm^T)$

$$
\begin{aligned}
&\partial_c(Jm^T) = \\
&\sum_j \frac{e_6(\theta_j^T)(JN_j^{T^t} \cdot N_j^T)_{(c)} N_j^T \cdot N_j^{T^t} \cdot JN_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^5} \\
&- \sum_j \frac{e_7(\theta_j^T)(2\eta - p_{t_{j+1}} - p_{t_{j+2}})_{(c)} N_j^T \cdot N_j^{T^t} \cdot JN_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^4} \\
&+ \sum_j \frac{e_1(\theta_j^T)\partial_c(N_j^T) \cdot N_j^{T^t} \cdot JN_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} + \sum_j \frac{e_1(\theta_j^T) N_j^T \cdot \partial_c(N_j^T)^t \cdot JN_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} \\
&- \sum_j \frac{3e_1(\theta_j^T)(\eta - p_{t_{j+1}})_{(c)} N_j^T \cdot N_j^{T^t} \cdot JN_j^T}{2|p_{t_{j+2}} - \eta|^3|p_{t_{j+1}} - \eta|^5} - \sum_j \frac{3e_1(\theta_j^T)(\eta - p_{t_{j+2}})_{(c)} N_j^T \cdot N_j^{T^t} \cdot JN_j^T}{2|p_{t_{j+2}} - \eta|^5|p_{t_{j+1}} - \eta|^3} \\
&- \sum_j \frac{\partial_c(N_j^T) \cdot (2\eta - p_{t_{j+1}} - p_{t_{j+2}})^t}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^2} + \sum_j \frac{(\eta - p_{t_{j+1}})_{(c)} N_j^T \cdot (2\eta - p_{t_{j+1}} - p_{t_{j+2}})^t}{|p_{t_{j+2}} - \eta|^2|p_{t_{j+1}} - \eta|^4} \\
&+ \sum_j \frac{(\eta - p_{t_{j+2}})_{(c)} N_j^T \cdot (2\eta - p_{t_{j+1}} - p_{t_{j+2}})^t}{|p_{t_{j+2}} - \eta|^4|p_{t_{j+1}} - \eta|^2} + \sum_j \frac{e_8(\theta_j^T)(JN_j^{T^t} \cdot N_j^T)_{(c)} JN_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} \\
&- \sum_j \frac{e_9(\theta_j^T)(2\eta - p_{t_{j+1}} - p_{t_{j+2}})_{(c)} JN_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^2} - \sum_j \frac{(\eta - p_{t_{j+1}})_{(c)} e_2(\theta_j^T) JN_j^T}{2|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|^3} \\
&- \sum_j \frac{(\eta - p_{t_{j+2}})_{(c)} e_2(\theta_j^T) JN_j^T}{2|p_{t_{j+2}} - \eta|^3|p_{t_{j+1}} - \eta|} - \sum_j \frac{N_j^T \cdot \delta^{ct}}{|p_{t_{j+2}} - \eta|^2|p_{t_{j+1}} - \eta|^2}
\end{aligned}
$$

with $e_6(x) = e_1'(x)cos(x)/sin(x), e_7(x) = e_1'(x)sin(x), e_8(x) = e_2'(x)cos(x)/sin(x)$, and $e_9(x) = e_2'(x)sin(x)$.

As previously discussed, $|N_j^T|$ can become close to 0 only if θ_j^T tends to 0 or π. The second case corresponds to η lying on one edge $p_{t_{j+1}}p_{t_{j+2}}$ of the triangle T. As discussed in section 10.3, we do not provide expressions of the derivatives in that case (for points lying on T).

The first case corresponds to η lying on the same line as one edge of the triangle. All the degenerate functions in this expression admit well-defined Taylor expansions. As these expressions are shown to converge, they provide a practical way to robustly evaluate $\partial_c(Jm^T)$ near the support lines of the edges of the cage triangles.

When η lies exactly onto the support plane of a triangle T, we cannot use the same strategy to compute the Hessians. These special cases will be discussed in the next paragraph.

Final expression of Hw_i^T

From Eq. 10.24, once again we obtain

$$\left\{ \begin{array}{ll} \partial_x(\overrightarrow{\bigtriangledown} w_{t_i}^T) & = \frac{C_x^{T^t} \cdot N_i^T}{\det(A^T)} \\ \partial_y(\overrightarrow{\bigtriangledown} w_{t_i}^T) & = \frac{C_y^{T^t} \cdot N_i^T}{\det(A^T)} \\ \partial_z(\overrightarrow{\bigtriangledown} w_{t_i}^T) & = \frac{C_z^{T^t} \cdot N_i^T}{\det(A^T)} \end{array} \right\} \quad \forall \eta \notin Support(T).$$

Since

$$Hw_i^T = \left(\begin{array}{c} \partial_x(\overrightarrow{\bigtriangledown} w_{t_i}^T)^t \\ \partial_y(\overrightarrow{\bigtriangledown} w_{t_i}^T)^t \\ \partial_z(\overrightarrow{\bigtriangledown} w_{t_i}^T)^t \end{array} \right)$$

finally we obtain

$$\boxed{\begin{array}{r} Hw_i^T = \frac{1}{\det(A^T)} \left(\begin{array}{c} N_i^{T^t} \cdot \partial_x(Jm^T) \\ N_i^{T^t} \cdot \partial_y(Jm^T) \\ N_i^{T^t} \cdot \partial_z(Jm^T) \end{array} \right) \\ + \frac{1}{\det(A^T)} (N_i^T \cdot (\sum_j \overrightarrow{\bigtriangledown} w_j^T)^t + \sum_j \overrightarrow{\bigtriangledown} w_j^T \cdot N_i^{T^t}) \end{array}} \quad (10.25)$$

Special case: $\eta \in Support(T), \notin T$

For all $\eta \in Support(T), \notin T$, Eq. 10.25 does not hold anymore, as A^T is a degenerate basis. Using $\overrightarrow{\bigtriangledown} w_i^T = dw_i^T n_T$ with $dw_i^T(\eta) = \frac{\partial(w_i^T(\eta + \epsilon n_T))}{\partial \epsilon}_{|\epsilon \to 0}$ (previously expressed in section 10.5.1), we obtain:

$$\boxed{Hw_i^T = \overrightarrow{\bigtriangledown} dw_i^T \cdot n_T^t \quad , \quad \forall \eta \in Support(T), \notin T} \quad (10.26)$$

The details of the derivation of $\overrightarrow{\bigtriangledown} dw_i^T$ can be found in Appendix B (see section 13.2). The final expression of $\overrightarrow{\bigtriangledown} dw_i^T$ is given by:

$$
\begin{aligned}
&-2|T|\overrightarrow{\bigtriangledown} dw_i^T = \\
&-\sum_j \frac{((p_{t_{i+2}} - p_{t_{i+1}})^t \cdot (p_{t_{j+2}} - p_{t_{j+1}}))(2\eta - p_{t_{j+2}} - p_{t_{j+1}})}{(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^2} \\
&+\sum_j \frac{e_1(\theta_j^T)((p_{t_{i+2}} - p_{t_{i+1}})^t \cdot (p_{t_{j+2}} - p_{t_{j+1}}))JN_j^{T^t} \cdot N_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} \\
&+\sum_j \frac{|p_{t_{j+2}} - p_{t_{j+1}}|^2(N_i^{T^t} \cdot N_j^T)(2\eta - p_{t_{j+2}} - p_{t_{j+1}})}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^4} \\
&+\sum_j \frac{e_1(\theta_j^T)|p_{t_{j+2}} - p_{t_{j+1}}|^2(JN_j^{T^t} \cdot N_i^T + JN_i^{T^t} \cdot N_j^T)}{4(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} \\
&-\sum_j \frac{e_4(\theta_j^T)|p_{t_{j+2}} - p_{t_{j+1}}|^2(N_i^{T^t} \cdot N_j^T)JN_j^{T^t} \cdot N_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^5} \\
&+\sum_j \frac{\cos(\theta_j^T)e_3(\theta_j^T)(JN_j^{T^t} \cdot N_i^T + JN_i^{T^t} \cdot N_j^T)}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^2} \\
&-\sum_j \frac{(1 - 2\cos(\theta_j^T))(N_i^{T^t} \cdot N_j^T)(2\eta - p_{t_{j+2}} - p_{t_{j+1}})}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} \\
&+\sum_j \frac{e_5(\theta_j^T)(N_i^{T^t} \cdot N_j^T)JN_j^{T^t} \cdot N_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^4}
\end{aligned}
$$

where $e_1(x) = \frac{\cos(x)\sin(x)-x}{\sin(x)^3}, e_4(x) = \frac{2\cos(x)\sin(x)^3+3(\sin(x)\cos(x)-x)}{\sin(x)^5}, e_3(x) = \frac{\cos(x)-1}{\sin(x)^2}$, and $e_5(x) = \frac{\cos(x)\sin(x)^2(1-2\cos(x))-2\cos(x)^2+2\cos(x)}{\sin(x)^4}$ are functions well defined on $]0, \pi[$ and that admit controllable Taylor expansion formula around 0.

10.6 Continuity between the general case and the special case

We obtained the formulae for the gradient and the Hessian of $w_i^T(\eta)$ in the general case, when the point of interest η does not lie on the triangle T, and in the special case when η lies on it.

As MVC are C^∞ everywhere not on $\mathbf{M}$, these formulae are guaranteed to converge, since in particular, the gradient and the Hessian are continuous functions everywhere not on $\mathbf{M}$.

The same holds in 2D where the distinction is made for the computation of $w_i^E(\eta)$ whether η lies on the line supported by the edge E or not.

10.7 Experimental Analysis

In this section, we present experimental evidence of the numerical accuracy of our derivation and provide computation timings.

10.7.1 Complexity

For each point η, computing the MVC, the MVC gradients and the MVC Hessians is linear in the number of vertices and edges (faces in 3D) of the cage.

10.7.2 Implementation

In this chapter, we have introduced several functions that were noted $e_k(x)$. These functions are unstable near 0, but all have clean Taylor expansions around 0 (their exact expression are of no scientific value, but they can be obtained using standard numerical computing environments, e. g. Matlab). In our implementation, we rely on these Taylor expansions up to the order 15 for the computation of $e_k(x)$ for all values of x smaller than 0.1, and consider the true expressions of the functions otherwise.

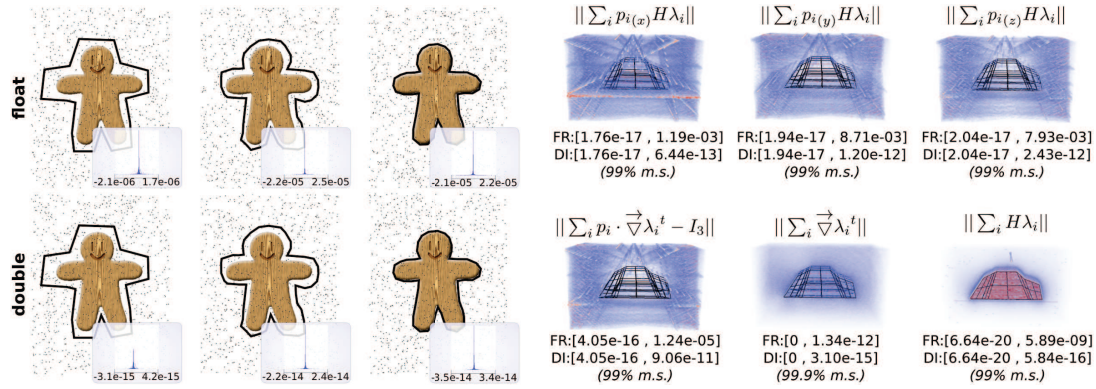
This strategy prooved to be efficient for their computation. However, they can be approximated with any precision using look-up tables, themselves computed using an *infinite precision* library[1].

10.7.3 Global validation with a manufactured solution

We first inspect the numerical accuracy of our derivation through the Method of Manufactured Solution (MMS), a popular technique in code verification [5, 32, 33]. Such a verification approach consists of designing an input configuration such that the resulting solution is known a priori. Then the actual verification procedure aims at assessing that the solution provided by the program conforms to the manufactured solution. In other words, MMS verification consists of the design of *exact ground-truths* for accuracy measurement. However, note that this verification is not general, as it only assesses correctness for the set of manufactured solutions.

Manufactured solution: As Mean Value Coordinates provide smooth interpolations, a global rigid transformation of the cage $\overline{p_i} = T + R \cdot p_i$ should infer a global rigid transformation of the entire Euclidean space. In particular, the Jacobians of the embedding function f of the transformed cage ($f : \mathbb{R}^3 \to \mathbb{R}^3$, $f(p_i) = \overline{p_i}$) should be equal to R everywhere, and its Hessian should be exactly 0. Then our manufactured configuration is the space of global rigid transformations and our manufactured solution is defined by $Jf = R$ and $Hf = 0$.

[1] For instance, see the GNU Multiple Precision Arithmetic Library, http://gmplib.org/



(a) 2D (b) 3D

Figure 10.3: **Validation based on a manufactured solution (group of rigid transformations)**
Left (a: 2D case): The histograms show the violation of the correctness conditions associated with the manufactured solution (95% most relevant samples). Top: simple floating point precision (output precision: 10^{-5}). Bottom row: double precision (output precision: 10^{-14}). Size of the diagonal of the domain: ~ 1000.
Right (b: 3D case): The violation of the correctness conditions (from transparent blue, low values, to opaque red, high value) is measured on each vertex of a 100^3 voxel grid. The full value range (FR) is given below each image while only the 99% most significant samples are displayed (DI). Size of the diagonal of the domain: $\sim$600.

Given this manufactured solution, we can derive correctness conditions for the Jacobian evaluation from the following expression:

$$Jf = \sum_i f(p_i) \cdot \vec{\nabla}\lambda_i{}^t = R \cdot \sum_i p_i \cdot \vec{\nabla}\lambda_i{}^t + T \cdot \sum_i \vec{\nabla}\lambda_i{}^t$$

Thus, to conform to the manufactured solution $Jf = R$, the following equations should be satisfied:

$$\begin{cases} \sum_i \vec{\nabla}\lambda_i{}^t = (0,0,0) \\ \sum_i p_i \cdot \vec{\nabla}\lambda_i{}^t = I_3 \end{cases} \tag{10.27}$$

As for the Hessian evaluation, we can derive similar correctness conditions:

$$\begin{aligned} Hf_c &= \sum_i f_c(p_i) H\lambda_i \\ &= \sum_{\forall d \in \{x,y,z\}} R_{cd} \cdot \sum_i p_{i(d)} H\lambda_i + T_c \sum_i H\lambda_i \quad \forall c = \{x,y,z\} \end{aligned}$$

where T_x, T_y and T_z are the first, second and third coordinate of the vector T respectively (similarly for R_{cd}).

Thus, to conform to the manufactured solution $Hf = 0$, the following equations should be satisfied:

$$\begin{cases} \sum_i H\lambda_i = 0_3 \\ \sum_i p_{i(x)} H\lambda_i = 0_3 \\ \sum_i p_{i(y)} H\lambda_i = 0_3 \\ \sum_i p_{i(z)} H\lambda_i = 0_3 \end{cases} \tag{10.28}$$

Note that both the Jacobian and Hessian correctness conditions (Eq. 10.27 and 10.28) are not functions of the rigid transformation parameters. These properties remain valid for arbitrary translations and rotations and thus cover the entire group of rigid transformations.

Fig. 10.3(a) shows numerical evaluations of these correctness conditions for different cages, at random points (in grey) of a 2D domain. In particular, the histograms plot the entries of the left-hand term (a vector or a matrix) of each of these equations, which should all be zero (for the second Jacobian condition, the entries of the matrix $\sum_i p_i \cdot \overrightarrow{\bigtriangledown}{\lambda_i}^t - I_2$ are shown). As shown in this experiment, the error induced by the violation of the correctness conditions is close to the actual precision of the data structure employed for real numbers (*float* or *double*). Also, the error slightly increases when the cage is denser. Indeed, with dense cages, it is more likely that the randomly selected samples lie in the vicinity of the support of the cage edges. These configurations correspond to the special cases discussed earlier and for which Taylor expansions are employed.

Fig. 10.3(b) shows a similar experiment in 3D, with a coarse cage (black lines). Similarly, most of the errors are located on the tangent planes of the triangles (special cases). Note, that an important part of the error yields from the samples which are located in the vicinity of the cage triangle, a configuration for which we do not provide a closed-form expression, as discussed in section 10.3. Interestingly, the errors on the correctness conditions for the Jacobian and the Hessian are comparable to the errors induced by the actual computation of the Mean Value Coordinates λ_i. In the example shown in Fig. 10.3(b), the error range of the positional reconstruction on the grid (i. e. the violation of the linear precision property of the MVC) is $[0, 1.08\ 10^{-5}]$, which is larger than the error ranges observed in two of the six correctness condition evaluations of the derivatives.

10.7.4 Taylor approximations behavior

Validation based on manufactured solutions enables assessing the accuracy of a numerical computation on a sub-set of pre-defined configurations. However, in our setting, designing manufactured solutions corresponding to other configurations than rigid transformations is highly involved.

Thus, to extend our analysis to arbitrary configurations, we present in this paragraph an analysis of the Taylor approximations of MVC-functions based on our derivation.

In contrast to manufactured solutions, this analysis is not meant to *validate* our results, but simply analyse how functions expressed by MVC behave in the local neighborhood of a point.

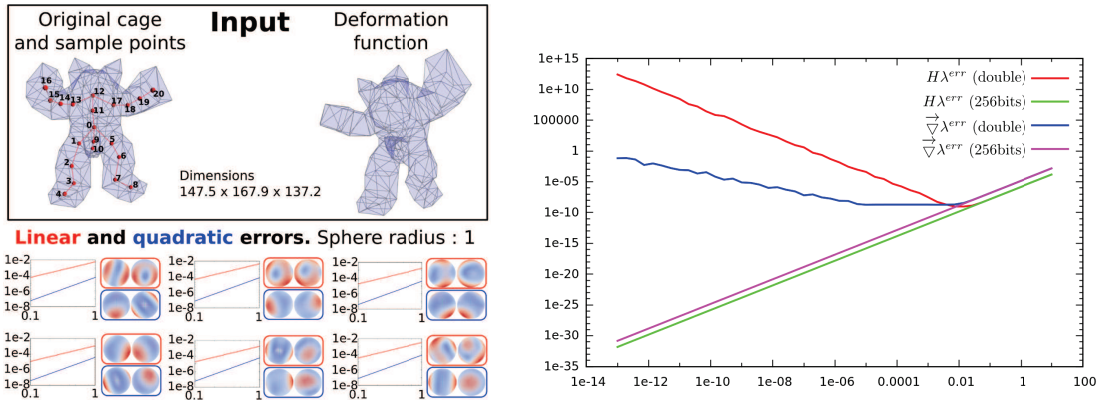


Figure 10.4: **Left:** Red curve: Linear approximation. Blue curve: Quadratic approximation. Plots show **on a logarithmic scale** the radial function defined as the average of the absolute error on the sphere of radius r ($r \in [0, 1]$). The points η where the evaluation was performed were taken on the skeleton shown in the original cage (here are displayed the evaluations for the points $\#0, \#1, \#2, \#3, \#4, \#5$, but these are representative of the curves we have for all locations). For each plot, the spheres on the top show the linear approximation error from two points of view (red box), the others show the quadratic approximation error from the same two points of view (blue box). From the tangent, we can assess the quadratic convergence of the linear approximation scheme and the cubic convergence of the quadratic approximation scheme.
Right: Comparison with Finite Differences. The domain are the same as described previously in Fig. 10.4(a), and the evaluations are performed on point 0. x axis: size of the stencil for Finite Differences. y axis: difference between Finite Differences approximations of the derivatives and our formulae. Axes of the plots are in logarithmic scale. The functions that are plotted are $\overrightarrow{\nabla}\lambda^{err}(r) = \sqrt{\sum_i ||\overrightarrow{\nabla}\lambda_i - \overrightarrow{\nabla}\lambda_i{}^{FD(r)}||^2}$ and $H\lambda^{err}(r) = \sqrt{\sum_i ||H\lambda_i - H\lambda_i{}^{FD(r)}||^2}$.

For regular functions, function values in the neighborhood of a point can be approximated up to several orders of precision, using Taylor approximations:

$$f(\eta + d\eta) = f(\eta) + \overrightarrow{\nabla} f_\eta^t \cdot d\eta + o(||d\eta||)$$

$$f(\eta + d\eta) = f(\eta) + \overrightarrow{\nabla} f_\eta^t \cdot d\eta + \frac{1}{2} d\eta^t \cdot Hf_\eta \cdot d\eta + o(||d\eta||^2)$$

In the following, we use these approximations to analyze the behavior of our derivation for arbitrary configurations. In particular, we evaluate the following errors:

$$E^1 = ||f(\eta + d\eta) - f(\eta) - \overrightarrow{\nabla} f_\eta^t \cdot d\eta||$$
$$E^2 = ||f(\eta + d\eta) - f(\eta) - \overrightarrow{\nabla} f_\eta^t \cdot d\eta - \frac{1}{2} d\eta^t \cdot Hf_\eta \cdot d\eta||$$

As the evaluation neighborhood shrinks to a point, these errors should tend to zero, with a horizontal tangent. Fig. 10.4(a) shows plots of these errors (logarithmic scale) on an arbitrary deformation function defined by user interactions:

Input Cage Model (#V / #T)	Coord. Only (ms)	Coord. + Deriv. [Analytic] (ms)	Coord. + Deriv. [FD] (ms)	Coord. + Deriv. [TriC] (ms)
Beetle (32 / 60)	0.060	0.781	1.157	4.196
Beetle (130 / 256)	0.256	3.261	4.881	17.625
Beetle (514 / 1024)	1.027	13.171	19.620	70.768
Armadillo (164 / 324)	0.324	4.130	6.174	22.428
Monster (128 / 252)	0.252	3.212	4.809	17.503
Monster (506 / 1008)	1.013	12.860	19.389	69.900

Table 10.1: Performance of the computation of the 3D Mean Value Coordinates and their derivatives at a single point. Tests were performed on 1000 points and average timings are presented. We compare our formulae computation time with Finite Differences (FD) methods that require 27 evaluations in total, and with tricubic approximations (TriC) that require 65 evaluations in total. Without any regards to the quality of such approximations, it is still faster to compute the true value of the derivatives using our formulae than approximating them using any Finite Differences scheme.

- On regular functions, the derivatives of the MVC characterize the interpolated function correctly: the maximum error is 7. 10^{-3} (for a bounding diagonal of 316). Note, that the radius $r \in [0, 1]$ of the evaluation neighborhood where they can be used to approximate the function is not too small. Therefore, our derivative formulation provides enough accuracy to enforce *sparse* derivative constraints on local neighborhoods such as those expressed in the applications described in section 10.8.

- The linear approximation can sometimes produce more accurate approximations in average than the quadratic approximation on a large neighborhood, while the quadratic approximation provides results which are less direction-dependant.

- The induced errors indeed tend to zero when the neighborhood shrinks to a point and the tangent of the curves in logarithmic scale illustrates that the error conforms to the expected form ($O(||d\eta||^2)$ for the linear approximation,$O(||d\eta||^3)$ for the quadratic approximation). Indeed, remember, that if $y = \lambda \cdot x^n$, then $\log(y) = \log(\lambda) + n \cdot \log(x)$.

10.7.5 Comparison with Finite Difference schemes

In this section we use Finite Differences schemes to derive the gradient and the Hessian of the MVC, to compare with the expressions we obtained.

A conventional scheme for approximations of first and second order derivatives is the following:

$f_x(x, y, z) \simeq \frac{f(x+h,y,z)-f(x-h,y,z)}{2h}$

$f_{xx}(x, y, z) \simeq \frac{f(x+h,y,z)-2f(x,y,z)+f(x-h,y,z)}{h^2}$

$f_{xy}(x, y, z) \simeq \frac{f(x+h,y+h,z)-f(x+h,y-h,z)-f(x-h,y+h,z)+f(x-h,y-h,z)}{4h^2}$

...

This scheme requires 27 evaluations of the function in total. Results of convergence of Finite Differences (FD) of the Mean Value Coordinates derivatives using this scheme are

presented on an example on Fig. 10.4(b), using double precision and 256 bits precision (using **mpfrc++**, which is a c++ wrapper of the *GNU multiple precision floating point library (mpfr)*). The domain is the same as described previously in Fig. 10.4(a), and the plots correspond here to the evaluation made in *point* 0. The error functions that are plotted are $\overrightarrow{\bigtriangledown}\lambda^{err}(r) = \sqrt{\sum_i ||\overrightarrow{\bigtriangledown}\lambda_i - \overrightarrow{\bigtriangledown}{\lambda_i}^{FD(r)}||^2}$ and $H\lambda^{err}(r) = \sqrt{\sum_i ||H\lambda_i - H{\lambda_i}^{FD(r)}||^2}$. Note, that these plots are representative of all the experiments we made (i. e. with other cages, at other locations, etc.).

These results validate empirically our formulae, as the Finite Differences scheme converges to our formulae when the size of the stencil tends to 0 (Fig. 10.4(b), using 256 bits precision). It also indicates that Finite Differences schemes are not suited to evaluate MVC derivatives in real life applications (see Fig. 10.8 for example), as these schemes diverge near 0 when using double precision only (Fig. 10.4(b) blue and red curves).

Note also, that we used different schemes to approximate the derivatives using Finite Differences methods (9 points evaluation + linear system inversion, 27 points evaluation on a $3 \times 3 \times 3$-stencil, tricubic interpolation on a $4 \times 4 \times 4$-stencil), and that they all diverge in the same manner when using double precision.

The error curves are also similar when looking at the deviation of the gradients and Hessians of the **function itself** that is interpolated (e. g. the deformation function in Fig. 10.4(a)), instead of the gradients and Hessians of the **weights** themselves.

10.7.6 Timings

Table 10.1 shows average computation times of the evaluation of the Mean Value Coordinates and their derivatives for several input cages. As the cost of the evaluation depends on the occurence of the special cases (point lying on the support plane of the triangles of the cage), we performed the computation on a set of 1000 points that were randomly distributed inside the bounding box of the model, and the average time is presented. As shown in this table, these computations take only a few milliseconds, which allows their usage in interactive contexts. Also, note that in the applications discussed in the following section, the derivatives are only evaluated on a very small set of points for constraint enforcement.

10.8 Applications

In this section, we review the applications presented in the original paper [47], and illustrate the utility of our contribution for all of them.

10.8.1 MVC derivatives visualization

In the context of function design/editing/visualization, the derivatives of the function can be of use to the user, as they have very often an intuitive meaning.

Figure 10.5: Visualization of rotations on the shape skeleton.

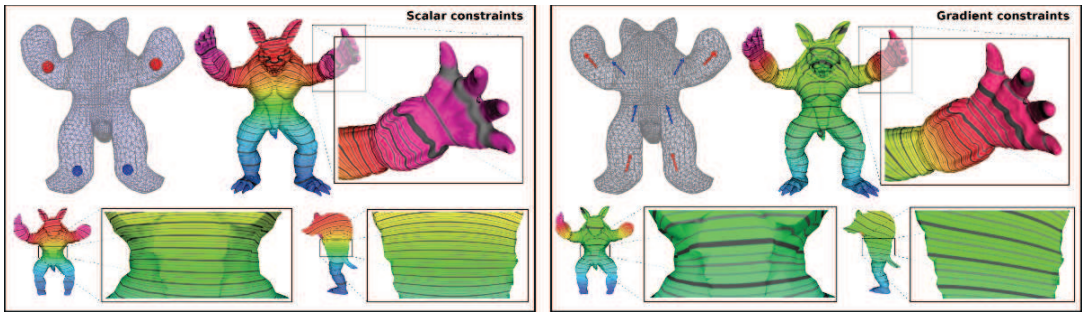


Figure 10.6: Flexible volumetric scalar field design with MVC gradient constraints. Left: Scalar constraints (spheres in the cage). Right: Gradient constraints (arrows in the cage). Blue and red colors respectively correspond to low and high value/gradient. In contrast to simple scalar constraints (left), gradient constraints (right) enable to intuitively interact with the shape and the velocity of the level lines.

For example, in the context of 2D or 3D deformation, the Jacobian of the transformation $J(\eta)$ can be put in the form $J(\eta) = R(\eta) \cdot B(\eta) \cdot \Sigma(\eta) \cdot B(\eta)^t$ using Singular Value Decomposition. These different matrices represent the scales of the transformation (Σ) in the basis given by B, and the rotation that is applied afterwards (R) – which can be represented easily as a vector and an angle (see Fig. 10.5). In the context of color interpolation, the gradients of the different channels can be displayed. In general, the norm of the Hessian provides the information of how locally rigid the function is around the point of interest. Using our formulation, one can obtain these informations at any scale with the same precision, to the contrary of what *finite differences schemes* would provide.

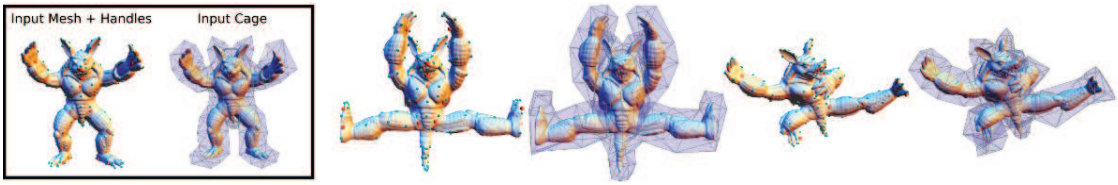


Figure 10.7: Implicit Cage Manipulation with *Variational MVC*. Red points indicate positional constraints; blue points indicate unknown rotational constraints. The quality of the produced cages allows the user to edit small details manually by switching from an *implicit* manipulation to an *explicit* manipulation. The same cannot be done with a Green Coordinates solver.

10.8.2 Flexible volumetric scalar field design

As shown in [47], Mean Value Coordinates can be used to solve the boundary value interpolation problem for the definition of volumetric scalar fields, given an input field prescribed on a closed surface. Our derivation of the MVC gradients and Hessians enables to extend this application to more flexible volumetric scalar field designs, in particular by enforcing the gradient of the interpolated function. Such flexible scalar fields contribute to volumetric texturing [47] and meshing [67]. Fig. 10.6 illustrates this application where the user sketched gradient constraints inside the volume. To compute a function which satisfies these constraints, a linear system is solved, where the unknowns are the scalar field values on the cage. In particular the following energy is minimized:

$$E = \sum_{v_i \in \overline{V}} w_i^P || \sum_j \lambda_j(v_i) f_j - \overline{f_i}||^2$$
$$+ \sum_{v_i \in \mathbf{M}} || \triangle f_i||^2 + \sum_{v_i \in \overline{G}} w_i^G || \sum_j \overrightarrow{\nabla} \lambda_j(v_i) \cdot f_j - \overline{g_i}||^2$$

where $\overline{V}$ is a set of points where hard constraints are applied on function values, $\triangle f_i$ denotes the cotangent Laplacian of the function at the vertex c_i of the cage, and $\overline{G}$ is the set of points where the gradient constraints are specified. Such an optimization procedure generates a smooth function on the cage (by minimizing its Laplacian) as well as in the interior volume (thanks to the MVC) with enforced gradient constraints. As shown in Fig. 10.6, the gradient constraints enable interacting with the shape and the *velocity* of the level sets of the designed function.

10.8.3 Implicit Cage Manipulation with Variational MVC

As shown in [8], intuitive volumetric deformations with little distortion can be obtained through a variational framework. In this context, the space of allowed transformations is explicitly described and the cage deformation is automatically optimized to satisfy positional constraints, while respecting the allowed transformations.

Since our derivation enables to express the Jacobian and Hessian of the transformation at any point in space as a linear combination of the cage vertices, the user can specify rigidity

constraints (by minimizing the norm of the Hessian) or rotational constraints (by setting the Jacobian to the corresponding value).

The solution to this optimization problem is a 3D field $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, defined everywhere in space using MVC, which interpolates the transformation of the cage vertices. We note $\overline{P}$ the set of positional constraints of the transformation ($\forall v_i \in \overline{P}, f(v_i) = \overline{v_i}$), $\overline{J}$ the set of Jacobian constraints ($\forall v_i \in \overline{J}, Jf(v_i) = \overline{J_i}$), and $\overline{H}$ the set of Hessian constraints ($\forall v_i \in \overline{H}, Hf_x(v_i) = Hf_y(v_i) = Hf_z(v_i) = 0_3$). The solution is given by minimizing the following energy:

$$\begin{aligned} E = &\sum_{v_i \in \overline{P}} (w_i^P || \sum_j \lambda_j(v_i) c_j - \overline{v_i}||^2) \\ + &\sum_{v_i \in \overline{J}} (w_i^J || \sum_j c_j \cdot \overrightarrow{\bigtriangledown} {\lambda_j}^t(v_i) - \overline{J_i}||^2) \\ + &\sum_{v_i \in \overline{H}} (w_i^H || \sum_j H\lambda_j(v_i) \cdot c_{j_{(x)}}||^2) \\ + &\sum_{v_i \in \overline{H}} (w_i^H || \sum_j H\lambda_j(v_i) \cdot c_{j_{(y)}}||^2) \\ + &\sum_{v_i \in \overline{H}} (w_i^H || \sum_j H\lambda_j(v_i) \cdot c_{j_{(z)}}||^2) \end{aligned}$$

where w^P, w^J, and w^H are weights for the positional, Jacobian, and Hessian constraints respectively. Similarly to [8], the transformation can be constrained locally to be a pure rotation. Then, in prescribed locations, the following property should hold:

$$Jf(v_i)^t \cdot Jf(v_i) = I_3 \quad \forall v_i \in \overline{J}$$

Note, that the actual values of these Jacobian constraints are now unknowns which can be obtained through an iterative optimization, as described in [8]. Due to the non-local nature of Mean Value Coordinates (in comparison to Green Coordinates), we constrain pure rotations to a subset of the enclosed surface vertices (blue spheres in Fig. 10.1 and 10.7) instead of constraining them to the medial axis.

Fig. 10.1 illustrates the algorithm, where the input surface is shown on the left in its enclosing cage. In this example, rotational constraints have been distributed evenly on the surface (blue points) and only 14 positional constraints have been specified and edited by the user. The interactions required by our system are limited and intuitive and our resolution of this optimization process is fast enough to provide interactive feedback despite a CPU-only implementation.

Fig. 10.8 shows a comparison with the results one can obtain using a Finite Differences scheme in the context of shape deformation. Using Finite Differences requires to tune the size of the stencil used for computation of the MVC derivatives case by case, and it can be difficult to set it up correctly, resulting in poor reconstructions.

Discussion

Other techniques have been proposed in the past for as-rigid-as-possible (ARAP) cage-driven shape deformations. For instance, Borosán et al. [11] presented a technique which

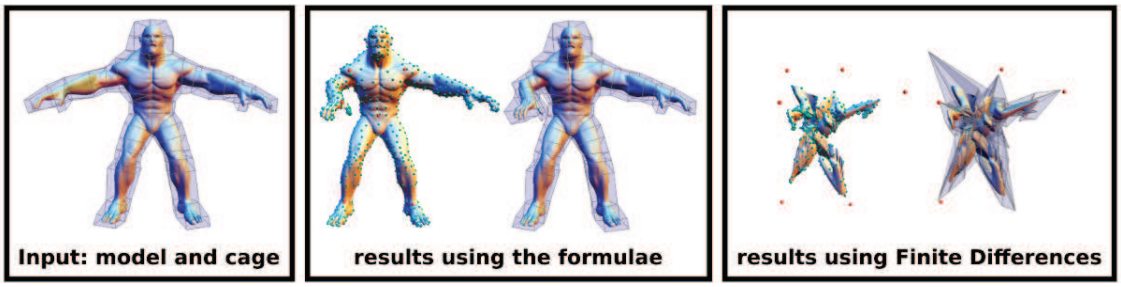


Figure 10.8: Comparison with Finite Differences schemes in the context of Variational MVC Deformations. The parameters for the linear system are strictly the same.

solves for ARAP transformations on the cage, while interpolating the results in the interior with MVC. However, as discussed earlier, MVC coordinates exhibit a very global behavior. Thus, ARAP transformations on the cage do not necessarily imply ARAP transformations in the interior. Reciprocally, it is often necessary to generate non-ARAP transformations of the cage in order to yield ARAP transformations in the interior. Instead, our technique enforces ARAP constraints directly on the enclosed shape.

In contrast to [8], our optimization process does not involve the normals of the triangles of the cage. Hence, it requires the resolution of fewer unknowns. Also, as the triangle normals are unknowns of the system in variational harmonic maps (VHM), they can take values arbitrarily far from the actual normals dictated by the Euclidean cross product of face edges. Hence, the cages generated by this technique cannot be exploited in a consistent manner for post-processing tasks. For instance, loading these cages in a modeling software supporting Green Coordinates would fail to correctly reconstruct the enclosed shape if the traditional Euclidean normals were used. Even if the normal solutions provided by the VHM system were used for this initial reconstruction, it would be not clear how to update them consistently given some explicit user deformation of the cage. The same remark goes for other post-processing tasks, such as cage-driven shape interpolation, for animation generation based on key-frames provided by implicit cage manipulation.

On the contrary, our technique (*variational MVC*) does not suffer from this drawback as only cage vertex positions are unknowns. Thus, the cages produced by our algorithm can be manipulated and re-used directly and consistently with existing software supporting MVC based deformations in various post-processing tasks. Also, our technique allows the user to switch at any time from *implicit* to *explicit* cage manipulation for small detail tuning, which cannot be done with a solver based on Green Coordinates. Note however, that it is not clear how to go from an *explicit* manipulation to an *implicit* manipulation, as the constraints enforced by our system are not respected when moving each cage's vertex independently from the others. Thus, *implicit* manipulation of the cage using our system can only be done before an *explicit* manipulation, or this editing phase will be discarded by the system.

It may be possible to express the derivatives for Positive Mean Value Coordinates (PMVC) [60] using our formula. The possible non-positivity of MVC coordinates has often been discussed as a drawback in certain contexts. Note however, that this particular

property makes them the only barycentric coordinates which allow the definition of coordinates outside of the cage in a straightforward manner. Nevertheless, PMVC can overcome this possible drawback, by only taking into consideration the cage vertices which are *visible* from the point under evaluation, which can be done very efficiently on the GPU, using the rasterization hardware machinery. However, these coordinates are not smooth since the visibility function is not smooth either. It would be interesting to study in practice how reliable the MVC derivatives can be when restricted to visibility dependant sub-cages, in order to mimic the behavior observed with PMVC. In the original paper, the authors motivate the approximation of the PMVC with the rendering of the basis functions on cube-maps by pointing out the fact that computing the visibility function is extremely difficult (it would require to re-mesh the part of the cage that is visible from the considered point, and then to compute the MVC w.r.t. this new cage). It seems difficult to do, but we recall that in most scenarios (e. g. the Variational MVC), the computation of the derivatives is required at a few locations only.

Chapter 11

Analytic biharmonic coordinates

We have seen in the technical background section of this thesis (see section 3.3.3), that 3D space transformations are limited to be at most quasi-conformal (or to belong to the class of Möbius transformations, which are too rigid to be considered in shape deformation tools). Surface-based transformation techniques allow to consider conformal deformations, but even those are quite limited, as they do not allow the twisting of objects (which is not conformal).

Biharmonic Coordinates have been made popular for shape deformation by Jacobson et al. [41], but their formulation does not come with a close formula, and the computation of this set of coordinates is quite involved. Although it was not the only contribution of this paper (it is probably more popular for the interaction process that is presented, that allows to manipulate at the same time *control cages, control skeletons, and control points*), it made the point that *non-harmonic* deformation functions can be of help to Computer Graphics artists.

We are interested in this work in finding a **close formula for the expression of *biharmonic* coordinates**, expressed w.r.t. a control cage of the form of a close triangular mesh.

11.1 Mathematical background

Deriving an equivalent of the Green's third identity in the biharmonic case, we express all biharmonic functions in terms of boundary constraints on the cage (when subdividing enough the cage surface, all functions can be expressed in this basis). We show, that expressing these constraints as continuous, piecewise linear for the second order and discontinuous, constant per triangle results in a set of coordinates that are appropriate for shape deformation. We make use of our set of coordinates in the context of shape deformation driven by a variational framework that optimizes for well considered sets of deformation spaces (limited to be smoothly varying by minimizing the norm of the Hessian of the function, and to be as-rigid-as-possible on the medial axis of the shape, resulting in quasi-

conformal deformations), and show their expression power. We also use this framework to compare our set of coordinates to previous work.

Contributions

In this chapter, we make the following contributions:

1. an introduction of our so-called Green BiHarmonic Coordinates, based on a piecewise linear 2^{nd} and constant per triangle 3^{rd} orders derivatives constraints on the cage (see section 11.2);
2. the close form expression of the integrals of the Green biharmonic functions on triangles;
3. a framework similar to Variational Harmonic Maps [8] that uses our Green BiHarmonic Coordinates, and that we call Variational BiHarmonic Maps (see section 11.3).

11.1.1 Laplace equations with boundary conditions

As presented in [61], a Green identity can be derived to express any harmonic function λ everywhere in a close domain D by the means of an integration of its boundary conditions:

$$\begin{aligned}\lambda(\eta) = &\int_{\xi\in\partial D} \lambda(\xi)\frac{\partial_1 G(\xi,\eta)}{\partial n_\xi}d\sigma_\xi \\ &- \int_{\xi\in\partial D} G(\xi,\eta)\frac{\partial \lambda(\xi)}{\partial n_\xi}d\sigma_\xi\end{aligned} \tag{11.1}$$

where G is the fundamental solution to the Laplace equation, i. e. $\triangle_1 G(x,y) = \triangle_2 G(x,y) = \delta_0(||x-y||)$.

Note that G is given in dimension d by

$$G(x,y) = \begin{cases} \frac{1}{(2-d)|S^d|}||x-y||^{2-d} & d \geq 3 \\ \frac{1}{2\pi}\log(||x-y||) & d = 2 \end{cases}$$

where $|S^d|$ denotes the volume of the unit sphere in $\mathbb{R}^d$.

In the same way, any **biharmonic** function λ can be expressed as

$$\begin{aligned}\lambda(\eta) = &\int_{\xi\in\partial D} \lambda(\xi)\frac{\partial_1 G(\xi,\eta)}{\partial n_\xi}d\sigma_\xi \\ &- \int_{\xi\in\partial D} G(\xi,\eta)\frac{\partial \lambda(\xi)}{\partial n_\xi}d\sigma_\xi \\ &+ \int_{\xi\in\partial D} \triangle\ \lambda(\xi)\frac{\partial_1 g(\xi,\eta)}{\partial n_\xi}d\sigma_\xi \\ &- \int_{\xi\in\partial D} g(\xi,\eta)\frac{\partial(\triangle\ \lambda)(\xi)}{\partial n_\xi}d\sigma_\xi\end{aligned} \tag{11.2}$$

where g is the fundamental solution to the biharmonic equation, i. e. $\triangle_1\ g(x,y) = \triangle_2\ g(x,y) = G(x,y)$, and $\triangle_1^2\ g(x,y) = \triangle_2^2\ g(x,y) = \delta_0(||x-y||)$. A simple proof of Eq. 11.2 can be found in Appendix D of this thesis (see section 13.4).

11.1.2 Green coordinates

In the following, we call *cage* a closed triangular mesh that acts as a control structure for shape deformation. Its vertices and their positions are similarly noted v_i, i being an integer index, and its triangles are noted $t_j = \{t_j^0, t_j^1, t_j^2\}$ and are represented as a triplet of the vertex indices of the triangle.

We consider a deformation function f defined on the cage (the boundary of the domain ∂D) that we wish to extend to the interior of the cage (the interior of the domain D). This function is necessarly piecewise linear on the cage. By noting $\Gamma_i(\xi)$ the piecewise linear basis function associated to the vertex v_i, that takes value 1 at v_i and 0 at other vertices, the function f can be expressed as:

$$\forall \xi \in \partial D \ : \ f(\xi) = \sum_{v_i} \Gamma_i(\xi) \cdot v_i \tag{11.3}$$

Lipman et al. [61] set the following Neumann conditions when deforming the cage:

$$\forall \xi \in t_j \subset \partial D \ : \ \frac{\partial f}{\partial n_\xi}(\xi) = s_{t_j} \cdot n(t_j) \tag{11.4}$$

with t_j being the j^{th} triangle of the cage, $n(t_j)$ being its outward unit normal, and s_{t_j} being the stretch factor associated to the deformation of the cage (its 2D conformality factor). As explained in the technical background of this thesis, this choice corresponds to setting the Jacobian of the function to be as close as possible to a similarity on each triangle of the cage.

The resulting deformation induced by the Green coordinates is expressed as

$$\begin{aligned} f(\eta) = &\sum_{v_i} \phi_i(\eta) \cdot v_i \\ &+ \sum_{t_j} \psi_j(\eta) s_{t_j} \cdot n(t_j) \end{aligned} \tag{11.5}$$

The functions $\phi_i(\eta)$ and $\psi_j(\eta)$ are the Green Coordinates. We refer the reader to [61] for an analysis of those, and to [8] for the expression of their derivatives.

Ultimately, we would like to have a similar argument that would allow us to set simple boundary constraints for our biharmonic functions, given the only information of the vertex positions. Unfortunately, this seems difficult, as the definition of higher order constraints requires more *global* information.

Biharmonic and harmonic functions If we set $\triangle\ f(\xi)$ and $\frac{\partial(\triangle f)(\xi)}{\partial n_\xi}$ to be null everywhere, we obtain the formulation proposed by [61]. Derived from our formulation, these

coordinates are still **biharmonic**. In fact, as explained in [61], they are **harmonic**, which is not in contradiction with our statement: a harmonic function is biharmonic (if $\triangle\ (f) = 0$, then $\triangle^2\ (f) = \triangle\ (\triangle\ (f)) = \triangle\ (0) = 0$).

Then again, if we consider a biharmonic function, its Laplacian is harmonic and therefore verifies the *maximum principle*. Setting $\triangle\ (f)$ to be null on the boundary of the domain should imply that it is null everywhere *inside* the domain, making harmonic the resulting function.

In the following sections, we discuss choices for the setting of other boundary conditions regarding the second order boundary condition ($\triangle\ (f)(\xi)$, see section 11.2) and the third order boundary condition ($\frac{\partial(\triangle f)(\xi)}{\partial n_\xi}$), and discuss the comparison with traditional harmonic Green coordinates in the context of cage-based shape deformations.

Note also, that **as for the Green coordinates**, the resulting function fits **none** of the boundary constraints. Indeed, to write an harmonic function, either a Dirichlet condition **or** a Neumann condition should be prescribed, and not both of them at the same time. You should note, that if the Dirichlet boundary constraint was respected, the resulting deformation would be the one given by the Harmonic Coordinates [46].

In the same way, a biharmonic function should be obtained by prescribing the Dirichlet condition and a Neumann condition, and no more boundary condition. The result of our formulation, as for the Green Coordinates, is the result of a mixture of these boundary conditions.

11.2 Green BiHarmonic coordinates: Analytic biharmonic cage-based deformations

Setting $\triangle\ (f)$ to be constant on each triangle gives biharmonic coordinates with a very simple expression. Unfortunately, we eventually found out that the behaviour induced by this set of coordinates was leading to severe distortion in the space deformation, making unusable these coordinates.

By setting $\triangle\ (f)$ to be linear on each triangle of the cage and continuous on the cage, and $\frac{\partial(\triangle f)(\xi)}{\partial n_\xi}$ to be constant on each triangle, i. e.

$$\begin{cases} \forall \xi \in \partial D \ : \ \triangle\ (f)(\xi) = \sum_{v_i} \Gamma_i(\xi) \cdot L_i^V \\ \forall \xi \in t_j \subset \partial D \ : \ \frac{\partial(\triangle f)(\xi)}{\partial n_\xi} = l_j^T \end{cases} \tag{11.6}$$

with $L_i^V, l_j^T \in \mathbb{R}^3$, we obtain additional coordinates w.r.t. the cage.

Finally, by noting $\Lambda_i^V(\eta) = \int_{\xi \in \{t \in T_1(v_i)\}} \Gamma_i(\xi) \frac{\partial_1 g(\xi,\eta)}{\partial n_\xi} d\sigma_\xi$ and $\lambda_j^T(\eta) = -\int_{\xi \in t_j} g(\xi,\eta) d\sigma_\xi$, we obtain that the deformation function f is expressed everywhere as

$$\begin{aligned} f(\eta) = &\sum_{v_i} \phi_i(\eta) \cdot v_i \\ &+ \sum_{t_j} \psi_j(\eta) \cdot n(t_j) \\ &+ \sum_{v_i} \Lambda_i^V(\eta) \cdot L_i^V \\ &+ \sum_{t_j} \lambda_j^T(\eta) \cdot l_j^T \end{aligned} \tag{11.7}$$

The coordinates ϕ_i and ψ_j are **harmonic** [61], and the coordinates Λ_i^V and λ_j^T are **biharmonic**.

The resulting function f is biharmonic as well, and

$$\begin{aligned} \triangle\,(f)(\eta) &= \textstyle\sum_{v_i} \triangle\,(\Lambda_i^V)(\eta) \cdot L_i^V + \sum_{t_j} \triangle\,(\lambda_j^T)(\eta) \cdot l_j^T \\ &= \textstyle\sum_{v_i} \phi_i(\eta) \cdot L_i^V + \sum_{t_j} \psi_j(\eta) \cdot l_j^T \end{aligned} \tag{11.8}$$

$$\triangle^2\,(f)(\eta) = 0 \tag{11.9}$$

The space of our biharmonic functions is a vectorial space of dimension $d_{BH}(\mathbf{M}) = 2(\#\mathcal{V} + \#\mathcal{T})$.

Motivations behind the choice of the constraints We chose to set $\triangle\,(f)$ to be continuous on the cage, as the Laplacian of a biharmonic function should be C^∞ everywhere inside the domain, and we consider its restriction on a 2-manifold of order of regularity C^0. We set it to be linear on each triangle as a default choice: it corresponds to the simplest basis function for continuous functions on triangular meshes.

Setting $\frac{\partial(\triangle f)(\xi)}{\partial n_\xi}$ to be discontinuous across the cage edges corresponds to the discontinuities of the normal on the cage (the normal is constant on each triangle, and $\frac{\partial(\triangle f)(\xi)}{\partial n_\xi} \triangleq \overrightarrow{\bigtriangledown} \triangle\, f(\xi) \cdot n_\xi$). We could have set this constraint to be linear on each triangle to enrich the space (providing each triangle with three values in $\mathbb{R}^3$ as a space to define it), but our experiments have shown that the behavior of this set of coordinates was introducing instabilities in the context of implicit cage-based deformations (see section 11.3).

For completeness, we provide the expression of $\Lambda_{j,k}^T(\eta) = \int_{\xi \in \{t_j\}} \Gamma_{t_j^k}(\xi) \frac{\partial_1 g(\xi,\eta)}{\partial n_\xi} d\sigma_\xi$ and $\lambda_{j,k}^T(\eta) = \int_{\xi \in \{t_j\}} \Gamma_{t_j^k}(\xi) g(\xi,\eta) d\sigma_\xi$.

Once these expressions are found, we can obtain $\Lambda_i^V(\eta)$ and $\lambda_j^T(\eta)$ as

$$\begin{cases} \Lambda_i^V(\eta) = \sum_{j,k|t_j^k=i} \Lambda_{j,k}^T(\eta) \\ \lambda_j^T(\eta) = \sum_k \lambda_{j,k}^T(\eta) \end{cases} \tag{11.10}$$

Convergence Note, that by subdividing $\mathbf{M}$, any function can be approximated using this basis for the constraints. This set of functions is therefore a good candidate for a basis of biharmonic functions, when using Finite Elements methods.

11.2.1 Computation of $\Lambda^T_{j,k}(\eta)$

Since $g(\xi,\eta) = \frac{||\xi-\eta||}{8\pi}$, we have that $\overrightarrow{\bigtriangledown}_\xi g(\xi,\eta) = \frac{(\xi-\eta)}{8\pi||\xi-\eta||}$, and therefore, $\frac{\partial_1 g(\xi,\eta)}{\partial n_\xi} = \frac{(\xi-\eta)^t\cdot n_\xi}{8\pi||\xi-\eta||} = \frac{(\xi-\eta)^t\cdot n(t_j)}{8\pi||\xi-\eta||}\forall\xi\in t_j$.

Since $\forall\xi\in t_j$, $\frac{\partial_1 g(\xi,\eta)}{\partial n_\xi} = \frac{(\xi-v_{t_j^0})^t\cdot n(t_j)}{8\pi||\xi-\eta||} + \frac{(v_{t_j^0}-\eta)^t\cdot n(t_j)}{8\pi||\xi-\eta||}$, we have finally, that

$$\forall\xi\in t_j, \frac{\partial_1 g(\xi,\eta)}{\partial n_\xi} = \frac{d(\eta,t_j)}{2}G(\xi,\eta) \tag{11.11}$$

, where $d(\eta,t_j)$ denotes the signed distance of η to the triangle t_j (i. e. $d(\eta,t_j) \triangleq (\eta - v_{t_j^0})^t\cdot n(t_j)$).

Using Eq. 11.11, we have
$\Lambda^T_{j,k}(\eta) = \int_{\xi\in t_j}\Gamma_{t_j^k}(\xi)\frac{\partial_1 g(\xi,\eta)}{\partial n_\xi}d\sigma_\xi = \frac{d(\eta,t_j)}{2}\int_{\xi\in t_j}\Gamma_{t_j^k}(\xi)G(\xi,\eta)d\sigma_\xi = \frac{d(\eta,t_j)}{2}\mu_{j,k}(\eta)$
where $\mu_{j,k}(\eta) = \int_{\xi\in t_j}\Gamma_{t_j^k}(\xi)G(\xi,\eta)d\sigma_\xi$.

Expression of $\mu_{j,k}(\eta)$ $\quad \forall\xi\in t_j\ ,\ \Gamma_{t_j^k}(\xi) = \frac{(\xi-v_{t_j^{k+1}})\cdot n_j^k}{(v_{t_j^k}-v_{t_j^{k+1}})\cdot n_j^k}$.

$$\mu_{j,k}(\eta) = \frac{\int_{\xi\in t_j}(\xi-\eta)\cdot G(\xi,\eta)d\sigma_\xi + \int_{\xi\in t_j}(\eta-v_{t_j^{k+1}})\cdot G(\xi,\eta)d\sigma_\xi}{\Delta_j}\cdot n_j^k$$

with $\Delta_j = (v_{t_j^k} - v_{t_j^{k+1}})\cdot n_j^k = 2Area(t_j)\forall k$.

$$\mu_{j,k}(\eta) = \frac{\int_{\xi\in t_j}(\xi-\eta)\cdot n_j^k\cdot G(\xi,\eta)d\sigma_\xi}{\Delta_j} + \frac{(\eta-v_{t_j^{k+1}})\cdot n_j^k\int_{\xi\in t_j}G(\xi,\eta)d\sigma_\xi}{\Delta_j}$$
$$= \frac{\int_{\xi\in t_j}(\xi-\eta)\cdot n_j^k\cdot G(\xi,\eta)d\sigma_\xi}{\Delta_j} + \frac{(\eta-v_{t_j^{k+1}})\cdot n_j^k\psi_j(\eta)}{\Delta_j}$$

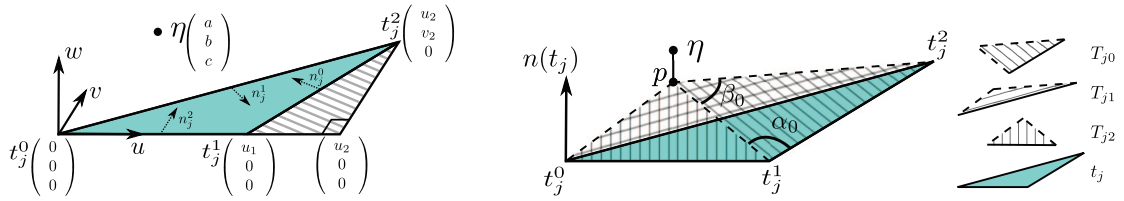
The expression $\frac{(\eta-v_{t_j^{k+1}})\cdot n_j^k}{\Delta_j}$ is the extension of the basis function associated with the vertex t_j^k of the cage $\Gamma_{t_j^k}$, from the triangle t_j to the entire 3D space. We note these functions $\tilde{\Gamma}_{t_j,k}(\eta)$, and they have the following properties:

$$\begin{cases} \sum_{k=0}^{2}\tilde{\Gamma}_{t_j,k}(\eta) & = 1 \quad \forall\eta\in\mathbb{R}^3 \quad \forall t_j \\ \overrightarrow{\bigtriangledown}\tilde{\Gamma}_{t_j,k}(\eta) & = \frac{n_j^k}{\Delta_j} \quad \forall\eta\in\mathbb{R}^3 \quad \forall t_j,k \end{cases} \tag{11.12}$$

All that remains is to find the expression of this last integral. We see that
$\forall\eta\ ,\ \int_{\xi\in t_j}(\xi-\eta)\cdot n_j^k\cdot G(\xi,\eta)d\sigma_\xi = \frac{-1}{4\pi}\cdot\int_{\xi\in t_j}\frac{(\xi-\eta)\cdot n_j^k}{||\xi-\eta||}d\sigma_\xi$.

We obtain

$$\mu_{j,k}(\eta) = \frac{-1}{4\pi\Delta_j}\cdot tf_{t_j,k}(\eta) + \tilde{\Gamma}_{t_j,k}(\eta)\psi_j(\eta)$$



(a) Notations in the local orthonormal basis defined in paragraph 11.2.1.

(b) Notations defined in paragraph 11.2.2. In this case $t_j = T_{j0} - T_{j1} + T_{j2}$.

Figure 11.1: Notations for the change of variables in integrals of paragraphs 11.2.1 and 11.2.2.

Expression of $tf_{t_j,k}(\eta)$ We make the following change of variables in this integral: $(u, v, w) = R \cdot (\xi - v_{t_j^0})$, with R being the rotation matrix $[\frac{v_{t_j^1} - v_{t_j^0}}{||v_{t_j^1} - v_{t_j^0}||}, n(t_j) \times \frac{v_{t_j^1} - v_{t_j^0}}{||v_{t_j^1} - v_{t_j^0}||}, n(t_j)]$ (see Fig. 11.1(a)).

$$tf_{t_j,k}(\eta) = \int_{\xi \in t_j} \frac{< (\xi - \eta) | n_j^k >}{||\xi - \eta||} d\sigma_\xi =$$
$$\int_{(u,v) \in R(t_j - v_{t_j^0})} \frac{< ((u,v,0) - (a,b,c)) | R \cdot n_j^k >}{||(u,v,0) - (a,b,c)||} dudv$$

To simplify, we note (u_i, v_i) the coordinates of $v_{t_j^i}$ in this new frame (note, that $u_0 = v_0 = v_1 = 0$). Since we only need the dot product of this integral and the vectors n_j^k that belong in the support plane of the triangle, we need to compute only the integrals over the two first coordinates u and v (note, that $R \cdot n_j^k = (-(v_{k+2} - v_{k+1}), (u_{k+2} - u_{k+1}), 0)$).

$$tfu_{t_j}(\eta) = \int_{(u,v) \in R(t_j - v_{t_j^0})} \frac{(u-a)}{||(u,v,0) - (a,b,c)||} dudv$$
$$= \int_{v=0}^{v_2} \int_{u=\frac{u_2}{v_2}v}^{\frac{u_2-u_1}{v_2}v+u_1} \frac{(u-a)}{\sqrt{(u-a)^2 + (v-b)^2 + c^2}} dudv$$
$$= \int_{v=0}^{v_2} \left[\sqrt{(u-a)^2 + (v-b)^2 + c^2} \right]_{u=\frac{u_2}{v_2}v}^{\frac{u_2-u_1}{v_2}v+u_1} dv$$

$$
\begin{aligned}
tfv_{t_j}(\eta) &= \int_{(u,v)\in R(t_j - v^0_{t_j})} \frac{(v-b)}{||(u,v,0)-(a,b,c)||} dudv \\
&= \int_{u=0}^{u_2} \int_{v=0}^{\frac{v_2}{u_2}u} \frac{(v-b)}{||(u,v,0)-(a,b,c)||} dvdu \\
&- \int_{u=u_1}^{u_2} \int_{v=0}^{\frac{v_2}{u_2-u_1}(u-u_1)} \frac{(v-b)}{||(u,v,0)-(a,b,c)||} dvdu \\
&= \int_{u=0}^{u_2} \left[\sqrt{(u-a)^2+(v-b)^2+c^2}\right]_{v=0}^{\frac{v_2}{u_2}u} du \\
&- \int_{u=u_1}^{u_2} \left[\sqrt{(u-a)^2+(v-b)^2+c^2}\right]_{v=0}^{\frac{v_2}{u_2-u_1}(u-u_1)} du
\end{aligned}
$$

We note $q(\beta, T-\alpha)$ the relevant antiderivative

$$
q(\beta, T-\alpha) = \int^T \sqrt{(t-\alpha)^2+\beta} dt = \int^{T-\alpha} \sqrt{t^2+\beta} dt
$$

with

$$
q(\beta, T) = \frac{\beta \cdot asinh(T/\sqrt{\beta}) + T \cdot \sqrt{T^2+\beta}}{2}
$$

The final expressions of $tfu_{t_j}(\eta)$ annd $tfv_{t_j}(\eta)$ are given by
$tfu_{t_j}(\eta) = -f_1 q(\lambda_1 v_2^2, v_2 - v_2 s_1) + f_1 q(\lambda_1 v_2^2, -v_2 s_1) + f_2 q(\lambda_2 v_2^2, v_2 - v_2 s_2) - f_2 q(\lambda_2 v_2^2, -v_2 s_2)$,
and $tfv_{t_j}(\eta) = f_3 q(\lambda_1 u_2^2, u_2 - u_2 s_1) - f_3 q(\lambda_1 u_2^2, -u_2 s_1) + q(b^2+c^2, -a) - q(b^2+c^2, u_1 - a)$
$- f_4 q((u_2-u_1)^2 \lambda_2, u_2 - u_s) + f_4 q((u_2-u_1)^2 \lambda_2, u_1 - u_s)$

with

$$
\left\{
\begin{array}{ll}
f_1 & = \sqrt{\frac{u_2^2+v_2^2}{v_2^2}} \\
f_2 & = \sqrt{\frac{(u_2-u_1)^2+v_2^2}{v_2^2}} \\
f_3 & = \sqrt{\frac{u_2^2+v_2^2}{u_2^2}} \\
f_4 & = \sqrt{\frac{(u_2-u_1)^2+v_2^2}{(u_2-u_1)^2}} \\
\lambda_1 & = \left(\frac{(av_2-bu_2)^2}{(u_2^2+v_2^2)^2} + \frac{c^2}{u_2^2+v_2^2}\right) \\
\lambda_2 & = \left(\frac{((a-u1)v_2-(u_2-u_1)b)^2}{((u_2-u_1)^2+v_2^2)^2} + \frac{c^2}{(u_2-u_1)^2+v_2^2}\right) \\
s_1 & = \frac{au_2+bv_2}{u_2^2+v_2^2} \\
s_2 & = \frac{((u_2-u_1)(a-u_1)+bv_2)}{(u_2-u_1)^2+v_2^2} \\
u_s & = \frac{v_2 b(u_2-u_1)+u_1 v_2^2+a(u_2-u_1)^2}{(u_2-u_1)^2+v_2^2}
\end{array}
\right.
$$

11.2.2 Computation of $\lambda^T_{j,k}(\eta)$

Following the same strategy as earlier,
$\lambda^T_{j,k}(\eta) = -\int_{\xi \in t_j} \Gamma_{t_j^k}(\xi) g(\xi, \eta) d\sigma_\xi$

$= \frac{-\int_{\xi\in t_j}(\xi-\eta)\cdot n_j^k||\xi-\eta||}{8\pi\Delta_j} - \frac{\tilde{\Gamma}_{t_j,k}(\eta)}{8\pi}\int_{\xi\in t_j}||\xi-\eta||d\sigma_\xi$
$= \frac{-\int_{\xi\in t_j}(\xi-\eta)\cdot n_j^k||\xi-\eta||}{8\pi\Delta_j} + \tilde{\Gamma}_{t_j,k}(\eta)\lambda_j^T(\eta)$

Note, that the computation of $\lambda_j^T(\eta)$ is required for the computation of $\lambda_{j,k}^T(\eta)$, and not the other way around (recall that $\lambda_j^T(\eta) = \sum_k \lambda_{j,k}^T(\eta)$). We still provide the expression of $\lambda_{j,k}^T(\eta)$ for completeness, but they are not our primary goal.

Computation of $-\int_{\xi\in t_j}(\xi-\eta)\cdot n_j^k||\xi-\eta||$ This integral is equal to $(v_{k+2}-v_{k+1})TFU_{t_j}(\eta) - (u_{k+2}-u_{k+1})TFV_{t_j}(\eta)$, with $TFU_{t_j}(\eta)$ and $TFV_{t_j}(\eta)$ the same expressions as $tfu_{t_j}(\eta)$ and $tfv_{t_j}(\eta)$, simply by replacing $q(\cdot,\cdot)$ in their expressions by $q_2(\cdot,\cdot)$ defined as:

$q_2(\beta,T) \triangleq \int^T \frac{(t^2+\beta)^{\frac{3}{2}}}{3}dt$
$= \frac{T}{12}(T^2+\beta)^{\frac{3}{2}} + \frac{\beta T}{8}\sqrt{T^2+\beta} + \frac{\beta^2}{8}asinh(\frac{T}{\sqrt{\beta}})$

Computation of $\int_{\xi\in t_j}||\xi-\eta||d\sigma_\xi$ The second integral is quite more challenging to compute. We cannot use the same change of variables as before, as it leads to over-complicated expressions that seem impossible to integrate. Instead, we propose to use the same change of variables as Lipman, in its computation of the GC [61](see Fig. 11.1(b)).

We note p the projection of η on the triangle t_j, and $T_{jk} = \{p, v_{t_j^{k+1}}, v_{t_j^{k+2}}\}$ (the super indices being modulo 2). It is easy to see that the integral of any function on the triangle t_j can be decomposed on the three triangles T_{j0}, T_{j1}, T_{j2} using the formula:

$$\int_{\xi\in t_j} f(\xi)d\sigma_\xi = \sum_{k=0}^{2} sign(T_{jk})\int_{\xi\in T_{jk}} f(\xi)d\sigma_\xi \tag{11.13}$$

where $sign(T_{jk})$ denotes the orientation of the triplet of T_{jk} ($sign(T_{jk}) \triangleq sign(det([v_{t_j^{k+1}} - p, v_{t_j^{k+2}} - p, n(t_j)]))$).

All we need to do is compute the integral of $||\xi-\eta||$ over the triangles T_{jk}. We note $c \triangleq ||\eta-p||^2, \alpha_k \triangleq \measuredangle(pv_{t_j^{k+1}}v_{t_j^{k+2}}), \beta_k \triangleq \measuredangle(v_{t_j^{k+2}}pv_{t_j^{k+1}}), \zeta_k \triangleq ||v_{t_j^{k+1}} - p||^2\sin^2(\alpha_k)$ and $\delta_k \triangleq \pi - \alpha_k$.

For convenience, in the following, we drop the index k in the notations.

$\int_{\xi\in T_{jk}}||\xi-\eta||d\sigma_\xi = \int_{\xi\in T_{jk}}\sqrt{c+||p-\xi||^2}d\sigma_\xi = \int_{\theta=0}^{\beta}\int_{r=0}^{R(\theta)}\sqrt{c+r^2}rdrd\theta = \int_{\theta=0}^{\beta}\frac{(c+R(\theta)^2)^{\frac{3}{2}}}{3}d\theta - \frac{\beta c^{\frac{3}{2}}}{3}$

with $R(\theta) = \frac{||v_{t_j^{k+1}}-p||\sin(\alpha)}{\sin(\pi-\alpha-\theta)}$.

$\int_{\theta=0}^{\beta}(c+R(\theta)^2)^{\frac{3}{2}}d\theta = \int_{\theta=\delta-\beta}^{\delta}(c+\frac{\zeta}{sin^2(\theta)})^{\frac{3}{2}}d\theta$

By setting $x = \tan(\theta)$ we see that the relevant antiderivative is $\int^{\Theta} (c + \frac{\zeta}{sin^2(\theta)})^{\frac{3}{2}} d\theta = \int^{\tan(\Theta)} (c + \frac{\zeta(1+x^2)}{x^2})^{\frac{3}{2}} \frac{dx}{1+x^2} = Q(\zeta, c, \tan(\Theta))$

where

$$Q(\zeta, c, x) = \frac{-sign(x)}{2}[(3\sqrt{\zeta}c + \zeta^{\frac{3}{2}}) \ln\left(2\frac{\zeta+\sqrt{\zeta}\sqrt{cx^2+\zeta(1+x^2)}}{|x|}\right) + \frac{\sqrt{cx^2+\zeta(1+x^2)}\zeta}{x^2} - 2c^{\frac{3}{2}} \arctan\left(\frac{\sqrt{cx^2+\zeta(1+x^2)}}{\sqrt{c}}\right)]$$

By making the remark that ζ depends on α ($\zeta_k = ||v_{t_j^{k+1}} - p||^2 \sin^2(\alpha_k)$), we can prove that these expressions are well defined in case $\tan(\theta) \to 0$ for $\theta = \delta$ or $\delta - \beta$. In that case, this integral tends to 0 (indeed, it corresponds to a triangle with one of its angles being equal to π, therefore with a null area).

At this point, we know how to compute $\int_{\xi \in T_{jk}} ||\xi - \eta|| d\sigma_\xi$, we can compute $\int_{\xi \in t_j} ||\xi - \eta|| d\sigma_\xi$ by using Eq. 11.13, and finally $\lambda_j^T(\eta) = \frac{-1}{8\pi} \int_{\xi \in t_j} ||\xi - \eta|| d\sigma_\xi$.

11.3 Variational BiHarmonic Maps

In this section, we present a comparison of Green Coordinates and Green BiHarmonic Coordinates, based on an *implicit* cage manipulation framework, as introduced in [8].

The idea is simple: we are going to set values for cage vertex positions, cage triangle normals, and eventually cage Laplacians as solution of a least-squares system, which associated energy models positional constraints, Jacobian constraints, and rigidity constraints.

This type of framework has already been presented in details in the previous chapter (see section 10.8.3) to compare our Variational Mean Value Coordinates to VHM [8]. We call our framework *Variational BiHarmonic Maps*.

To evaluate the quality of the underlying cage coordinates system, we plot the conformality factor of the deformation, as done in [8]. We compare our framework to VHM using this set of statistics.

We use finite difference schemes to approximate the derivatives of our coordinates, and use them in our implementation. However, closed formulae could be obtained by direct derivation of the presented equations.

Comparison with VHM and VMVC

In each example in this section, red spheres indicate positional constraints and green spheres the places where the user constraints the Jacobian to be eiter a pure rotation or a pure similarity, depending on the case. The weights that are given to each of the three energies (positional, rotational, and minimization of the Hessian) are the same when comparing our set of coordinates to others, to allow fair comparison.

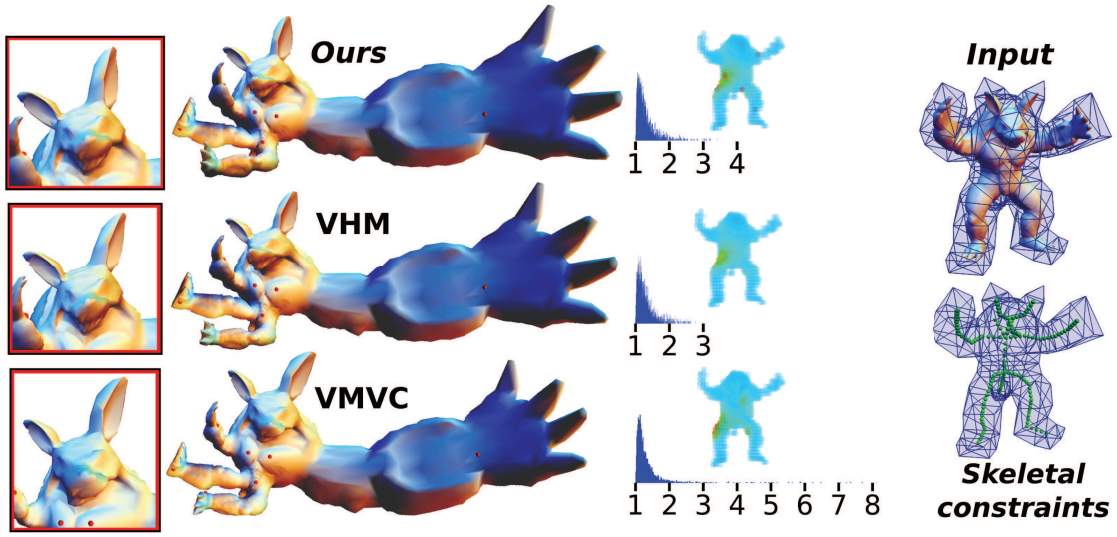


Figure 11.2: Comparison of our Variational BiHarmonic Maps (top row) with Variational Harmonic Maps based on Green coordinates (middle row) and Variational Mean Value Coordinates (bottom row).

We show a comparison of our system with VHM [8] and VMVC, introduced in the precedent chapter, on an example featuring extreme local scaling (*similarities* were enforced on the skeleton displayed in green, instead of rotations) in Fig. 11.2.

As already discussed in the previous chapter, MVC do not exhibit the mathematical properties to rely on this kind of setup, and rotational constraints as well as similarity constraints should be enforced directly on the target mesh. Some differences with VHM are visible on this example, although very subtile in our opinion. The conformality factors that are displayed correspond to the ones of the space transformation itself, and not to the transformation of the surface. They are computed as the condition number of the Jacobian of the transformation. In the case of our Variational BiHarmonic Maps, we see that some conformality factors seem to be unexpectedly high, especially when put in respect to the obtained deformation. We believe that this behavior appears because we rely on Finite Differences schemes to compute the derivatives of our set of coordinates. This introduces instabilities, as demonstrated in the previous chapter. Nevertheless, this point requires further investigation.

In Fig. 11.3, we show an example where rotations were enforced on the skeleton of the shape. The color code indicates the two dimensional conformality factor of each triangle (from 1 in blue, to 6 in red). It is computed as the ratio between the largest singular value σ_1 and the smallest singular value σ_2 of the 2×2 linear map that represents the deformation of the triangle in the plane. These quantities are classically used to evaluate the quality of a deformation of a triangle mesh; we refer the reader to [77] for a description of their computation.

We display in table 11.1 a set of statistics comparing our model to VHM and VMVC. E_{rot} computes the deviation of the mesh deformation from rigid deformations (featuring pure

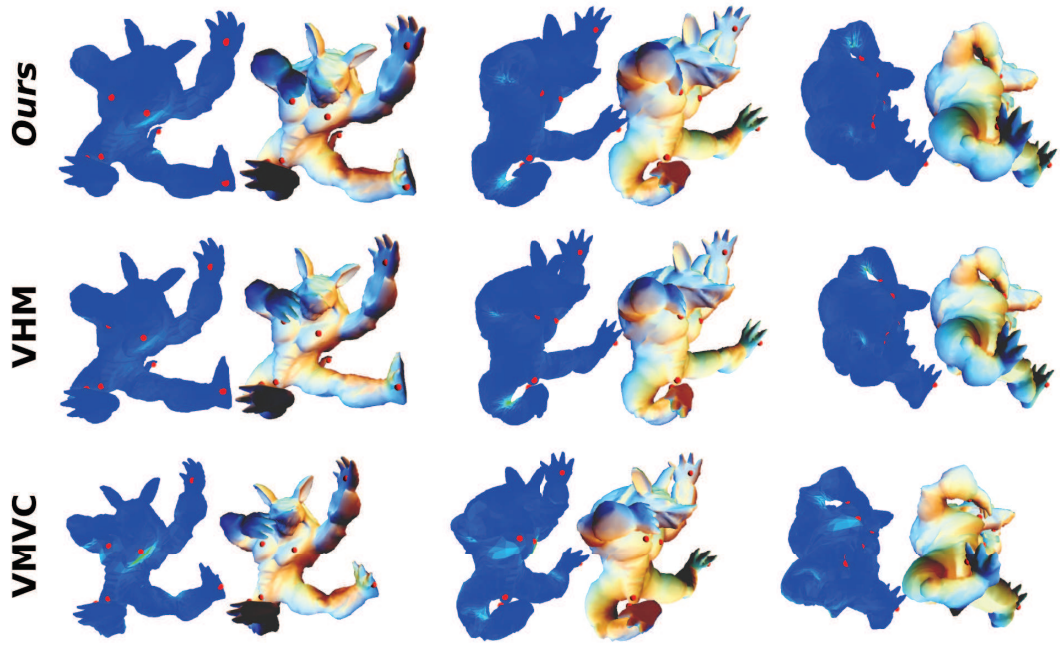


Figure 11.3: **Arma #1**: Comparison of our Variational BiHarmonic Maps (top row) with Variational Harmonic Maps based on Green coordinates (middle row) and Variational Mean Value Coordinates (bottom row), using the same setup as in Fig. 11.2.

Model	Ours			VHM			VMVC		
	R_{vol}	E_{sim}	E_{rot}	R_{vol}	E_{sim}	E_{rot}	R_{vol}	E_{sim}	E_{rot}
Arma #1	0.98790	0.02948	0.05020	0.95339	0.02977	0.05131	0.98364	0.05199	0.08113
Arma #2	1.02390	0.02457	0.03461	1.02027	0.02092	0.02907	1.02752	0.04372	0.06083
Arma #3	0.98754	0.04546	0.05753	0.97110	0.04016	0.05080	1.02045	0.06594	0.08556

Table 11.1: Deviation of the deformations presented in Fig. 11.3 and 11.4 from rigid deformations.

rotations everywhere), and E_{sim} the deviation of the mesh deformation from conformal deformations (featuring pure similarities everywhere). They are computed as

$$E_{rot} = \frac{\sum_{t_j \in \mathcal{T}} Area(t_j)((\sigma_1^j - 1)^2 + (\sigma_2^j - 1)^2)}{\sum_{t_j \in \mathcal{T}} Area(t_j)}$$

$$E_{sim} = \frac{\sum_{t_j \in \mathcal{T}} Area(t_j)(\sigma_1^j - \sigma_2^j)^2}{2 \sum_{t_j \in \mathcal{T}} Area(t_j)}$$

In addition to conformality errors, we also show the volume preservation ratio R_{vol} (ratio between the volume of the shape after and before the deformation) for the three systems. Enforcing rotations everywhere should lead to preservation of the volume (on the contrary to similarities, that constraints the deformation to be as conformal as possible). Our coordinates seem to exhibit statistics similar to VHM on the same data set. Recall that the variational framework that we use minimizes three kinds of energies: positional, rotational, and minimization of the Hessian. It may happen that the user specifies positional

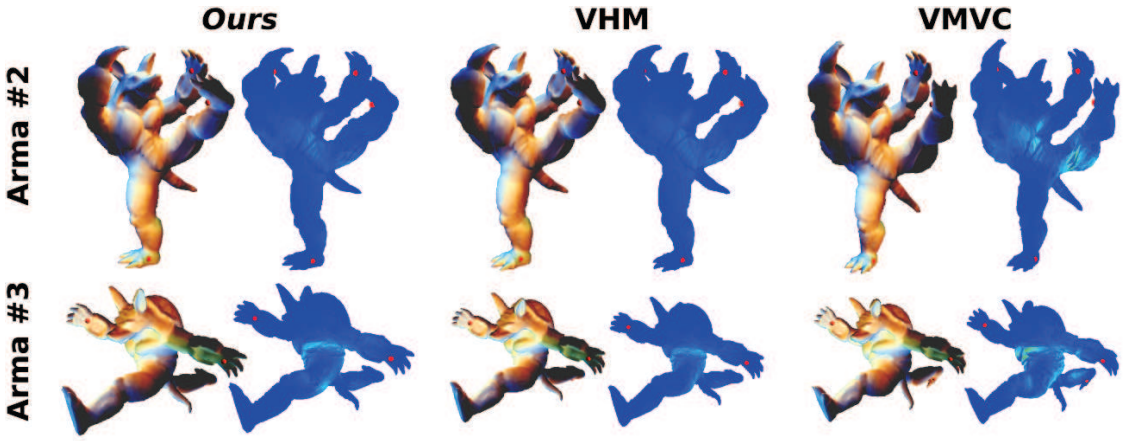


Figure 11.4: Comparison of our Variational BiHarmonic Maps (left) with Variational Harmonic Maps based on Green coordinates (middle) and Variational Mean Value Coordinates (right), using the same setup as in Fig. 11.2.

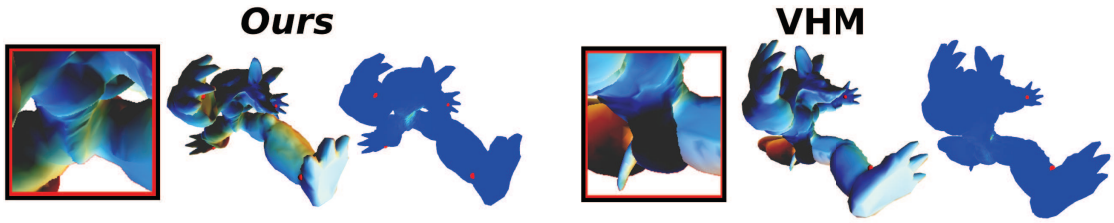


Figure 11.5: Self-intersections were avoided by our system.

constraints that are contradictory to the rigidity constraints on the shape. Although these statistics are interesting to quantify in some way the quality of the deformation methods, the visual quality of the deformations should prevail.

Fig. 11.5 shows another example with extreme scalings and rotations. Enforcing similarities on the medial axis is not always sufficient to obtain rigidity on the whole shape (and even avoid flip-overs). We prevent flip-overs on the medial axis by constraining the Jacobians to have a positive determinant. In some cases, it is still not enough to ensure that no flip-over or self-intersection are going to occur elsewhere. Using the same parameters for the linear system, the use of our biharmonic coordinates avoids self-intersections in that case, which seems to indicate that they allow quicker varying rotations in the shape. However, note that it is done at the cost of increasing slightly the similarity error ($E_{sim} = 0.01679$) in comparison to the solution given by VHM ($E_{sim} = 0.01002$). As already said in the introduction of this chapter, twisting introduces a loss of conformality.

11.4 Discussion

We have presented a set of analytic biharmonic coordinates in the context of shape deformation based on a variational framework. To our knowledge, the derived basis for biharmonic functions was not presented nor used before in the literature.

In all our examples, we used traditional positional constraints to drive the deformation. It would be interesting to see other kinds of constraints on this set of coordinates.

Other contexts than shape deformation may also benefit from a basis for biharmonic functions, although it is not clear how to retrieve values for the 2^{nd} and 3^{rd} orders of derivatives in general, and specific strategies would probably need to be used for each different case.

Finding a closed form formula for the derivatives of our biharmonic coordinates is of interest and such an additional analysis could precise more the values of the different statistics that we have shown in this chapter to compare our coordinates to previous methods. In particular, it may impact the quality of the deformation when using a variational framework that relies intensively on these quantities to drive the shape transformation.

Part VI

Conclusion and Perspectives

Chapter 12

Conclusion

In this thesis, we presented algorithms that aim at *extracting* simple structures that can drive complex computations on 3D objects. These structures can be geometrical or topological, can be located *inside* the shape, *on* the shape, and *outside* the shape. It can be *1*,*2*, or *3*-dimensional objects, or it can be a simple set of indices describing a subset of the object.

Based on the applicative constraints, we proposed each time new properties for the objects we were looking for, whether it is an *analytic* curve skeleton, a *simplicial complex* structure on the surface, or a *minimal set of handles* representing cage deformations. Furthemore, we introduced new geometry processing and analysis tools enabling the use of classical shape editing structures such as curve skeletons and cages in new applications.

Regarding *inner objects*, we proposed an analytic model for curve skeletons, along with an algorithm to find a segmentation of a mesh into topological cylinders and topological disks, from which the skeleton's geometry can be derived. The construction of this segmentation relies on a mesh decimation algorithm, that considers a 2-term cost function, whose first term describes a *topological event* in the clustering and second term describes a geometrical cost depending on the topology. To our knowledge, this graph segmentation problem was not introduced in the litterature before. In practice, it seems difficult to find a *global* strategy to solve it , as the geometrical energies that we minimize depend on the topology of the segmentation, and therefore it depends on the result itself. This opens a new graph segmentation problem that is interesting and general enough to allow new applications, even in other fields of research. The several properties of our skeleton model allow multiple applications, such as skeletal modeling and the definition of a new bilateral filter on triangular meshes, that supports large kernels and preserves mid-scale features.

Regarding *on-surface objects*, we proposed a *simplicial complex structure* as well as a *set of curves* on the surface, allowing the automatic definition of handles for mesh deformation techniques, thus relieving the artist from this time consuming task. The deformation complex is built in a multi-scale fashion upon a multi-resolution segmentation using standard partitioning systems, suggesting the selection of different kinds of handles (points, curves, patches) at different scales. The user interface allows to navigate easily through the different levels of the hierarchy, and to adapt the processing strategies to the nature

of the handle. The set of curves we propose is designed from intrinsic and view-dependent properties of the surface, and allows to use standard line rendering algorithms to define curve handles. We introduced an efficient strategy for the regularization of these curves and demonstrated its power in standard modeling sessions.

Regarding *outer objects*, we have demonstrated that an inversion based on *minimal selection of constraints* (MaxVol) improves the spectral properties of the linear system to inverse, and that our subsampling strategy coupled with subspectral regularization were adapted to the particular problem of finding a cage-based representation for animated 3D objects. Ultimately, the feedback offered by the solution to the MaxVol problem can drive the resampling of the cage object to represent a given geometry at its best. It opens an interesting avenue for finding an optimal sampling of the cage object, so that the cage-coordinate system offers optimal control over the deformed geometry.

Finally, we worked on the derivation of spatial coordinates, and obtained results that are of practical as well as theoretical value. We obtained an analytic solution for the derivatives of the mean value coordinates in 2D and in 3D, and demonstrated their utility on the applications that are commonly presented for them. We also obtained an analytic solution of biharmonic coordinates in 3D, and demonstrated their expressive power in the particular context of shape deformation. We have shown that their profile is adapted to implicit constraints set up on their first and second order derivatives, and presented an application to variational shape deformation.

Future work Throughout this thesis, we discovered some problematics that inspire us for future directions of research. Some of these directions are the direct continuation of the work we did during this thesis – such as designing automatic or user-assisted algorithms for creating optimal cages for mesh deformation, but some are complementary to our work, and describe orthogonal directions of research for interactive modeling.

Although we have seen that the mathematical derivation of cage coordinates for the definition of spatial functions has been intensively studied, we believe that *constructing the geometry of these cages* has not been solved so far. Various representations can be imagined to help designing cages: a hierarchy of enclosing spheres or enclosing boxes, etc. . Specific applications should benefit from the definition of such structures. These are numerous and various: volume simplification for fast intersection tests, volume abstraction, deformation, etc. . Different constraints need to be met, and not only the underlying geometry of the mesh has to be taken into account, but the deformation the artist wants to model as well. In this context, new ways of specifying these user-constraints have to be thought through, and specific interfaces need to be developed for the modeling of such volumetric objects.

The reverse engineering of the construction of 3D shapes has been studied recently. To our knowledge, it is limited to techniques allowing to discover a plausible succession of topological operations (e. g. local subdivision) to transform a coarse triangle "source" mesh into a high-resolution "target" mesh. One interesting avenue for future work is the *reverse engineering of the modeling process* of complex shapes. In particular, as described in the introduction of this thesis, 3D objects are usually designed by artists upon basic shapes described by simple primitives, such as boxes ("box modeling") or simple sets of connected spheres ("ZSPhere"-Pixologic). A clean, quad-dominant mesh can be obtained from these basic structures called "base meshes", on which the user can design high frequency details.

We believe that retrieving such high-level structures from complex geometrical objects can be done, and would allow new applications, such as the design of base meshes from example and more generally "modeling by example". It would also improve a large number of existing applications, such as remeshing for instance.

The definition of other types of "base meshes" is also a topic that we did not address in this thesis, and we are interested in designing such structures from scratch. These can be optimized to fit particular application needs, such as allowing easy definition of shape grammars on meshes for example, making possible the modeling from object data bases. New modeling interfaces would benefit from it. Specific constraints can be added on 3D objects in case they are designed to be re-used as modeling bricks and deformed in other 3D scenes – one can think about specifying simple descriptors to preserve the aspect ratio of important visual features, such as blazons or emblems, while allowing anisotropic bend and distortion on other parts of the object that can be performed by shape grammars to fit the editing domain. Recent work has been focused on painting geometry on surfaces. This geometry is generally taken from other surfaces in a data basis. This was the evolution of the classical brush-based techniques, that allow in most modeling frameworks to paint displacement or carving (negative displacement) on meshes. The displacement is generally not uniform, and simple grey scale textures can be used for the design of the motif of the displacement. The research direction we propose is the natural evolution of this concept: after having proposed uniform displacement brushes, anisotropic displacement brushes, and finally geometry brushes, one can think about defining object brushes, allowing the artist to paint a domain on a surface that acts as a support for the automatic arrangement of complex objects that need to be deformed using basic rules when being arranged.

In this thesis, we gave a particular importance to deformation and *motion*, which has been studied since the beginning of Computer Graphics. Recently in particular, new strategies have allowed to extract it from characters and rigid objects with *as-few-as-possible* assumptions on its nature. This allowed applications such as partial motion editing and motion transfer. Still, we feel that a lot of work remains for a proper definition of motion, that would allow retrieving it from data bases for example. In particular, recent work has demonstrated that the recognition of important features in static meshes involves both local and global descriptors. We believe that, when considering animated data, two more dimensions are going to sum up, as intrinsic geometry of the object and its temporal evolution (once decorrelated) should be both discriminant. Motion that is specific to certain classes of objects can give clues about the nature and the materials of the object, and should allow finer recognition of objects in animated scenes and in videos. This would imply the enrichment of the existing descriptors of motion on objects, as well as tracking algorithms in animated scenes when the mapping from one frame to another is not known, like in videos for instance.

Author's publications

1. **Curve Skeleton from Topological Disks & Cylinders Decomposition**
   ACM SIGGRAPH/Eurographics Symposium on Geometry Processing 2010 - Poster
   Jean-Marc Thiery, Tamy Boubekeur and Bert Buchholz

2. **Automatic Line Handles for Freeform Deformation**
   Eurographics 2012 - Short paper
   Leila Schemali, Jean-Marc Thiery and Tamy Boubekeur

3. **VoxMorph: 3-Scale Freeform Deformation of Large Voxel Grids**
   Computer & Graphics Journal 2012 - Special issue of Shape Modeling International 2012
   Noura Faraj, Jean-Marc Thiery and Tamy Boubekeur

4. **CageR: From 3D Performance Capture to Cage-based Representation**
   Siggraph 2012 - Talk
   Jean-Marc Thiery, Julien Tierny and Tamy Boubekeur

5. **CageR: Cage-based Reverse Engineering of Animated 3D Shapes**
   Computer Graphics Forum 2012
   Jean-Marc Thiery, Julien Tierny and Tamy Boubekeur

6. **Analytic Curve Skeletons for 3D Surface Modeling and Processing**
   Computer Graphics Forum 2012 - Special issue of Pacific Graphics 2012
   Jean-Marc Thiery, Bert Buchholz, Julien Tierny and Tamy Boubekeur

7. **Robust and Scalable Interactive Freeform Modeling of High Definition Medical Images**
   Springer Lecture Notes in Computer Science - Special issue of MICCAI Workshop on Mesh Processing in Medical Image Analysis 2012
   Noura Faraj, Jean-Marc Thiery, Isabelle Bloch, Nadège Varsier, Joe Wiart and Tamy Boubekeur

Data and software

We would like to thank the following organizations for providing the various models shown in this thesis:

- Aim@Shape Network
- Stanford University
- the research group 3D Video and Vision-based Graphics of the Max-Planck-Center for Visual Computing and Communication (MPI Informatik / Stanford)

We are particularly grateful to Tim Winkler and his colleagues for providing us with the implementation of their algorithm "Multi-Scale Geometry Interpolation" [98], as well as for the time they spent making it easy for us to use it.

Bibliography

[1] Blender: http://www.blender.org/.

[2] Zbrush: http://www.pixologic.com/zbrush/.

[3] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics (TOG). Special issue for SIGGRAPH conference*, 22(3):485–493, 2003.

[4] O.K.C. Au, C.L. Tai, H.K. Chu, D. Cohen-Or, and T.Y. Lee. Skeleton extraction by mesh contraction. In *ACM Transactions on Graphics (TOG)*, volume 27, page 44. ACM, 2008.

[5] I. Babuska and J. Oden. Verification and validation in computational engineering and science: basic concepts. *Computer Methods in Applied Mechanics and Engineering*, 193:4057–4066, 2004.

[6] C. Beder and W. Förstner. Direct solutions for computing cylinders from minimal sets of 3d points. pages 135–146. Springer, 2006.

[7] M. Ben-Chen, O. Weber, and C. Gotsman. Spatial deformation transfer. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 67–74. ACM, 2009.

[8] M. Ben-Chen, O. Weber, and C. Gotsman. Variational harmonic maps for space deformation. *ACM Transactions on Graphics (TOG)*, 28(3):34, 2009.

[9] H. Blum. A Transformation for Extracting New Descriptors of Shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[10] R. C. Bolles and M. A. Fischler. A ransac-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI'81: Proceedings of the 7th international joint conference on Artificial intelligence*, volume 2, pages 637–643, 1981.

[11] P. Borosán, R. Howard, S. Zhang, and A. Nealen. Hybrid Mesh Editing. In *Eurographics 2010-Short Papers*, pages 41–44.

[12] M. Botsch and L. Kobbelt. Multiresolution surface representation based on displacement volumes. 22(3):483–491, 2003.

[13] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 630–634. ACM, 2004.

[14] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[15] M. Botsch and O. Sorkine. On linear variational surface deformation methods. *Visualization and Computer Graphics, IEEE Transactions on*, 14(1):213–230, 2008.

[16] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005.

[17] T. Chaperon, F. Goulette, and C. Laurgeau. Extracting cylinders in full 3d data using a random sampling method and the gaussian image. In *Proceedings of the Vision Modeling and Visualization Conference*, pages 35–42. Citeseer, 2001.

[18] A. Civril and M. Magdon-Ismail. Finding maximum Volume sub-matrices of a matrix. *RPI Comp Sci Dept TR*, pages 1–13, 2007.

[19] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 905–914. ACM, 2004.

[20] N. D. Cornea, D. Silver, and P. Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007.

[21] K. Crane, U. Pinkall, and P. Schröder. Spin transformations of discrete surfaces. In *ACM Transactions on Graphics (TOG)*, volume 30, page 104. ACM, 2011.

[22] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, and H.P. Seidel. Performance capture from sparse multi-view video. *ACM Trans. Graph. (SIGGRAPH)*, 27, 2008.

[23] E. de Aguiar, C. Theobalt, S. Thrun, and Seidel H.-P. Automatic conversion of mesh animations into skeleton-based animations. *Comp. Graph. Forum (Eurographics)*, 27, 2008.

[24] K. G. Der, R. W. Sumner, and J. Popovic. Inverse kinematics for reduced deformable models. *ACM Trans. Graph. (SIGGRAPH)*, 25:1174–1179, 2006.

[25] M. Desbrun, E. Kanso, and Y. Tong. Discrete differential forms for computational modeling. *Discrete differential geometry*, pages 287–324, 2008.

[26] R. Dyer, H. Zhang, and T. Möller. Surface sampling and the intrinsic Voronoi diagram. In *ACM Symposium on Geometry Processing*, pages 1393–1402, 2008.

[27] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511–533, 2002.

[28] H. Edelsbrunner, D. Morozov, and V. Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 127–134. ACM, 2006.

[29] H. Edelsbrunner and N. R. Shah. Triangulating topological spaces. In *Proc. of the 10th Symposium on Computational Geometry*, pages 285–292, 1994.

[30] M. Eigensatz, R.W. Sumner, and M. Pauly. Curvature-domain shape processing. *Comp. Graph. Forum (Eurographics)*, 27:241–250, 2008.

[31] A. Ēivril and M. Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47-49):4801–4811, 2009.

[32] T. Etiene, L.G. Nonato, C. Scheiddeger, J. Tierny, T. J. Peters, V. Pascucci, , R.M. Kirby, and C.T. Silva. Topology verification for isosurface extraction. *IEEE Trans. on Vis. and Comp. Grap.*, 2011.

[33] T. Etiene, C. Scheiddeger, L.G. Nonato, R.M. Kirby, and C.T. Silva. Verifiable visualization for isosurface extraction. 15:1227–1234, 2009.

[34] M. S. Floater, G. Kos, and M. Reimers. Mean value coordinates in 3D. *Comp. Aided Geom. Design*, 22:623–631, 2005.

[35] A. Fomenko and T.L. Kunii. Topological modeling for visualization, 1997.

[36] J. Gain and D. Bechmann. A survey of spatial deformation from a user-centered perspective. *ACM Transactions on Graphics (TOG)*, 27(4):107, 2008.

[37] M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[38] S.A. Goreinov, L.V. Oaeledets, D.V. Savostyanov, E.E. Tyrtyshnikov, and N.L. Zamarashkin. How to find a good submatrix. *Matrix Methods: Theory, Algorithms and Applications*, page 247, 2010.

[39] P.C. Hansen, T. Sekii, and H. Shibahashi. The modified truncated svd method for regularization in general form. *SIAM Journal on Scientific and Statistical Computing*, 13:1142, 1992.

[40] H. Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108. ACM, 1996.

[41] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 30(4):78:1–78:8, 2011.

[42] A. Jacobson and O. Sorkine. Stretchable and twistable bones for skeletal shape deformation. *ACM Trans. Graph. (SIGGRAPH Asia)*, 2011.

[43] D.L. James and C.D. Twigg. Skinning mesh animations. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 399–407. ACM, 2005.

[44] S. Jin, R.R. Lewis, and D. West. A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21(1):71–82, 2005.

[45] T. R. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. In *ACM SIGGRAPH*, 2003.

[46] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. *ACM Trans. Graph. (SIGGRAPH)*, 26, 2007.

[47] T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. (SIGGRAPH)*, 24(3):561–566, 2005.

[48] T. Judd, F. Durand, and E. Adelson. Apparent ridges for line drawing. *ACM Transactions on Graphics (TOG)*, 26(3):19, 2007.

[49] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658, 2005.

[50] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 954–961, 2003.

[51] L. Kavan, P.P. Sloan, and C. O'Sullivan. Fast and efficient skinning of animated meshes. In *Computer Graphics Forum*, volume 29, pages 327–336. Wiley Online Library, 2010.

[52] M. Kilian, N.J. Mitra, and H. Pottmann. Geometric modeling in shape space. *ACM Transactions on Graphics (TOG)*, 26(3):64–es, 2007.

[53] S. Kircher and M. Garland. Editing arbitrarily deforming surface animations. *ACM Trans. Graph. (SIGGRAPH)*, 25:1098–1107, 2006.

[54] S. Kircher and M. Garland. Free-form motion processing. *ACM Trans. Graph.*, 27, 2008.

[55] V. Krayevoy and A. Sheffer. Variational, meaningful shape decomposition. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 50, 2006.

[56] T.-Y Lee, Y.-S. Wang, and T.-G. Chen. Segmenting a deforming mesh into near-rigid components. *The Visual Computer (Pacific Graphics)*, 22:729–739, 2006.

[57] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371, 2002.

[58] J. Liouville. Extension au cas des trois dimensions de la question du tracé géographique. *Application de l'Analysea la Géométrie*, pages 609–616, 1850.

[59] Y. Lipman, V.G. Kim, and T.A. Funkhouser. Simple formulas for quasiconformal plane deformations. *ACM Transactions on Graphics (TOG)*, 31(5):124, 2012.

[60] Y. Lipman, J. Kopf, D. Cohen-Or, and D. Levin. Gpu-assisted positive mean value coordinates for mesh deformations. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 117–123. Eurographics Association, 2007.

[61] Y. Lipman, D. Levin, and D. Cohen-Or. Green coordinates. *ACM Trans. Graph. (SIGGRAPH)*, 27(3):1–10, 2008.

[62] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, D. Rossl, and H.-P. Seidel. Differential coordinates for interactive mesh editing. In *IEEE Shape Modeling International*, pages 181–190, 2004.

[63] Y. Lipman, O. Sorkine, D. Levin, and D. Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. (SIGGRAPH)*, 24:479–487, 2005.

[64] Y. Liu, Z. Chen, and K. Tang. Construction of iso-contours, bisectors and voronoi diagrams on triangulated surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (99):1–1, 2011.

[65] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[66] M. Meyer, M. Desbrun, P. Schröder, and A.H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics*, 3(7):34–57, 2002.

[67] M. Nieser, U. Reitebuch, and K. Polthier. CubeCover - Parameterization of 3D volumes. *Comp. Graph. Forum (SGP)*, 30:1397–1406, 2011.

[68] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 58, 2007.

[69] U. Pinkall, , and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[70] T. Popa, I. South-Dickinson, D. Bradley, A. Sheffer, and W. Heidrich. Globally consistent space-time reconstruction. *Comp. Graph. Forum (SGP)*, 29:1633–1642, 2010.

[71] S. D. Porumbescu, B. C. Budge, Z. Feng, and K. Joy. Shell maps. *ACM SIGGRAPH 2005, ACM Transactions on Graphics*, 24(3):626–633, 2005.

[72] H. Pottmann. *Architectural geometry*, volume 10. Bentley Institute Press, 2007.

[73] T. Rabbani and F.A. van den Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. pages xx–yy, 2005.

[74] G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique on the singular points of a complete integral pfaff form or of a numerical function. *Comptes Rendus Acad. Science Paris.*, v222.:847–849, 1946.

[75] M. Reuter, S. Biasotti, D. Giorgi, G. Patanè, and M. Spagnuolo. Discrete laplace-beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33:381–390, 2009.

[76] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission*, September 2004.

[77] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416, 2001.

[78] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 146–155, 2003.

[79] Y. Savoye and J.-S. Franco. Cage-based tracking for performance animation. In *Asian Conference on Computer Vision*, 2010.

[80] S. Schaefer and C. Yuksel. Example-based skeleton extraction. In *Symposium on Geometry Processing*, pages 153–162, 2007.

[81] T.W. Sederberg and S.R. Parry. Free-form deformation of solid geometric models. In *ACM Siggraph Computer Graphics*, volume 20, pages 151–160. ACM, 1986.

[82] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[83] A. Sharf, T. Lewiner, A. Shamir, and L. Kobbelt. On-the-fly curve-skeleton computation for 3d shapes. *Computer Graphics Forum*, 26(3):323–328, September 2007.

[84] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, pages 109–116, 2007.

[85] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 355–362, 2002.

[86] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184. ACM, 2004.

[87] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popovic. Mesh-based inverse kinematics. *ACM Trans. Graph. (SIGGRAPH)*, 24:488–495, 2005.

[88] R.W. Sumner and J. Popović. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 399–405. ACM, 2004.

[89] A. Tagliasacchi, H. Zhang, and D. Cohen-Or. Curve skeleton extraction from incomplete point cloud. *ACM Transactions on Graphics, (Proceedings SIGGRAPH 2009)*, 28(3):Article 71, 9 pages, 2009.

[90] J.-M. Thiery, J. Tierny, and T. Boubekeur. Jacobians and Hessians of Mean Value Coordinates for Closed Triangular Meshes. Technical report, CNRS LTCI Telecom ParisTech, 2011.

[91] J. Tierny and V. Pascucci. Generalized topological simplification of scalar fields on surfaces.

[92] J. Tierny, J.-P. Vandeborre, and M. Daoudi. Enhancing 3d mesh topological skeletons with discrete contour constrictions. *The Visual Computer*, 24:155–172, 2008.

[93] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, page 836–846, 1998.

[94] T. Tung and F. Schmitt. Augmented reeb graphs for content-based retrieval of 3d mesh models. In *SMI '04: Proceedings of the Shape Modeling International 2004*, pages 157–166, 2004.

[95] M. Urago. Analytical integrals of fundamental solution of three-dimensional laplace equation and their gradients. *Trans. of the Japan Soc. of Mech. Eng.*, 66:254–261, 2000.

[96] O. Weber, M. Ben-Chen, and C. Gotsman. Complex barycentric coordinates with applications to planar shape deformation. *Computer Graphics Forum (Proceedings of Eurographics)*, 28(2), 2009.

[97] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. Context-aware skeletal shape deformation. *Comp. Graph. Forum (Eurographics)*, pages 265–274, 2007.

[98] T. Winkler, J. Drieseberg, M. Alexa, and K. Hormann. Multi-scale geometry interpolation. *Comp. Graph. Forum (Eurographics)*, 29:309–318, 2010.

[99] J. Wu and L. Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3):277–284, 2005.

[100] W. Xu, K. Zhou, Y. Yu, Q. Tan, Q. Peng, and B. Guo. Gradient domain editing of deforming sequences. *ACM Trans. Graph. (SIGGRAPH)*, 26, 2007.

[101] D.M. Yan, Y. Liu, and W. Wang. Quadric surface extraction by variational shape approximation. *Geometric Modeling and Processing-GMP 2006*, pages 73–86, 2006.

[102] X. Yang, A. Somasekharan, and J. J. Zhang. Curve skeleton skinning for human and creature characters: Research articles. *Comput. Animat. Virtual Worlds*, 17(3-4):281–292, 2006.

[103] S. Yoshizawa, A. Belyaev, and H.P. Seidel. Skeleton-based variational mesh deformations. *Comp. Graph. Forum (Eurographics)*, pages 255–264, 2007.

[104] Y. Zheng and C.L. Tai. Mesh decomposition with cross-boundary brushes. In *Computer Graphics Forum*, volume 29, pages 527–535. Wiley Online Library, 2010.

Chapter 13

Appendix

13.1 Appendix A: Taylor expansion of $w_{t_i}^T(\eta + \epsilon n_T)$

We present here the details for the Taylor expansion to the order 1 of $w_{t_i}^T(\eta + \epsilon n_T) = \frac{N_i^T(\eta+\epsilon n_T)^t \cdot m^T(\eta+\epsilon n_T)}{\det(A^T(\eta+\epsilon n_T))}$, in the special case where $\eta \in Support(T), \notin T$.

The determinant of a basis of $\mathbb{R}^3$ is the volume of its generated parallelepiped, and the following expression holds: $\det(A^T(\eta + \epsilon n_T)) = -2\epsilon \cdot |T|$, where $|T|$ is the area of T. All that is left is to write the Taylor expansion of $N_i^T(\eta + \epsilon n_T)^t \cdot m^T(\eta + \epsilon n_T)$ to the order 2.

$$\begin{aligned} &N_i^T(\eta + \epsilon n_T)^t \cdot m^T(\eta + \epsilon n_T) \\ &\quad = \frac{1}{2} \sum_j \theta_j(\eta + \epsilon n_T) \cdot N_i^T(\eta + \epsilon n_T)^t \cdot n_j^T(\eta + \epsilon n_T) \\ &\quad = \frac{1}{2} \sum_j \theta_j(\eta + \epsilon n_T) \cdot N_i^T(\eta + \epsilon n_T)^t \cdot \frac{N_j^T(\eta + \epsilon n_T)}{|N_j^T(\eta + \epsilon n_T)|} \end{aligned}$$

We need to develop $\theta_j^T(\eta + \epsilon n_T)$ to the second order:

$$\theta_j^T(\eta + \epsilon n_T) = \theta_j + \epsilon \overrightarrow{\bigtriangledown} \theta_j^{T^t} \cdot n_T + \frac{\epsilon^2}{2} n_T^t \cdot H\theta_j^T \cdot n_T + o(\epsilon^2)$$

From the expression of $\overrightarrow{\bigtriangledown} C_j^T$, we know that $\overrightarrow{\bigtriangledown} C_j^{T^t} \cdot n_T = 0 \ \forall \ \eta \in Support(T)$. Therefore $\overrightarrow{\bigtriangledown} \theta_j^{T^t} \cdot n_T = 0 \ \forall \ \eta \in Support(T)$, and

$$\theta_j^T(\eta + \epsilon n_T) = \theta_j + \frac{\epsilon^2}{2} n_T^t \cdot H\theta_j^T \cdot n_T + o(\epsilon^2)$$

We develop $|N_j^T(\eta + \epsilon n_T)|^{-1}$ to the second order:

$$\begin{aligned}\frac{1}{|N_j^T(\eta+\epsilon n_T)|} =& \frac{1}{|N_j^T|} - \epsilon\frac{(JN_j^{T^t}\cdot N_j^T)^t\cdot n_T}{|N_j^T|^3}\\ &- \frac{\epsilon^2}{2}\frac{n_T^t\cdot(JN_j^{T^t}\cdot JN_j^T)\cdot n_T}{|N_j^T|^3}\\ &+ 3\frac{\epsilon^2}{2}\frac{n_T^t\cdot(JN_j^{T^t}\cdot N_j^T\cdot N_j^{T^t}\cdot JN_j^T)\cdot n_T}{|N_j^T|^5} + o(\epsilon^2)\end{aligned}$$

We can start simplifying this expression by noticing that $(JN_j^T \cdot N_j^T)^t \cdot n_T = 0$ and $n_T^t\cdot(JN_j^{T^t}\cdot JN_j^T)\cdot n_T = |JN_j^T\cdot n_T|^2 = |p_{t_{j+2}} - p_{t_{j+1}}|^2$.

Therefore the development of the function $\eta \to |N_j^T|^{-1}$ in the direction of n_T is equal to

$$\frac{1}{|N_j^T(\eta+\epsilon n_T)|} = \frac{1}{|N_j^T|} - \frac{\epsilon^2}{2}\frac{|p_{t_{i+2}} - p_{t_{i+1}}|^2}{|N_j^T|^3} + o(\epsilon^2) \tag{13.1}$$

$$\begin{aligned}&N_i^T(\eta+\epsilon n_T)^t\cdot m^T(\eta+\epsilon n_T)\\ &\quad=\frac{1}{2}(N_i^T+\epsilon JN_i^T\cdot n_T)^t\cdot\\ &\quad\sum_j (N_j^T+\epsilon JN_j^T\cdot n_T)(\theta_j+\frac{\epsilon^2}{2}n_T^t\cdot H\theta_j^T\cdot n_T)(\frac{1}{|N_j^T|}-\frac{\epsilon^2}{2}\frac{|p_{t_{i+2}}-p_{t_{i+1}}|^2}{|N_j^T|^3})\\ &\quad+o(\epsilon^2)\end{aligned}$$

We note that $(JN_i\cdot n_T)^t\cdot N_j = 0$ and $N_i^t\cdot(JN_j\cdot n_T) = 0$, since N_k and n_T are colinear $\forall k,\ \ \forall\eta\in Support(T)$.

We also note that

$$\begin{aligned}(JN_i\cdot n_T)^t\cdot(JN_j\cdot n_T) &= ((p_{t_{i+2}}-p_{t_{i+1}})\wedge n_T)^t\cdot((p_{t_{j+2}}-p_{t_{j+1}})\wedge n_T)\\ &= (p_{t_{i+2}}-p_{t_{i+1}})^t\cdot(p_{t_{j+2}}-p_{t_{j+1}})\end{aligned}$$

By developing all the terms, we see that all that is left is the order two (plus higher orders, that we do not write, since they are negligeable in front of ϵ^2).

$$\begin{aligned}N_i^T(\eta+\epsilon n_T)^t\cdot m^T(\eta+\epsilon n_T) =& \epsilon^2\sum_j\frac{\theta_j^T(p_{t_{i+2}}-p_{t_{i+1}})^t\cdot(p_{t_{j+2}}-p_{t_{j+1}})}{2|N_j^T|}\\ &+\epsilon^2\sum_j\frac{n_T^t\cdot H\theta_j^T\cdot n_T\cdot N_i^{T^t}\cdot N_j^T}{4|N_j^T|}\\ &-\epsilon^2\sum_j\frac{\theta_j^T|p_{t_{j+2}}-p_{t_{j+1}}|^2N_i^{T^t}\cdot N_j^T}{4|N_j^T|^3}+o(\epsilon^2)\end{aligned}$$

$$
\begin{aligned}
N_i^T(\eta+\epsilon n_T)^t \cdot m^T(\eta+\epsilon n_T) = \epsilon^2 \sum_j \frac{\theta_j^T (p_{t_{i+2}} - p_{t_{i+1}})^t \cdot (p_{t_{j+2}} - p_{t_{j+1}})}{2|N_j^T|} \\
+ \epsilon^2 \sum_j \frac{(|N_j^T|^2 {n_T}^t \cdot H\theta_j^T \cdot n_T - \theta_j^T |p_{t_{j+2}} - p_{t_{j+1}}|^2) \cdot {N_i^T}^t \cdot N_j^T}{4|N_j^T|^3} \\
+ o(\epsilon^2)
\end{aligned}
$$

This expression is not well defined when $\exists j \ / \ \theta_j^T = 0$. In fact, $|N_j^T| \sim \theta_j^T$ around 0, and therefore the term under the first sum tends to a constant when $\theta_j^T \to 0$.

By combining this expression with the one of $H\theta_j^T$, we will obtain the final expression of $\overrightarrow{\bigtriangledown} w_i^T$, and we will be able to prove that the j-th term under the second sum tends to 0 when $\theta_j^T \to 0$.

By derivating twice $C_j^T = \cos(\theta_j^T) = \frac{<p_{t_{j+1}} - \eta | p_{t_{j+2}} - \eta>}{|p_{t_{j+1}} - \eta||p_{t_{j+2}} - \eta|}$, we have that $-\cos(\theta_j^T) \cdot \overrightarrow{\bigtriangledown} \theta_j^T \cdot \overrightarrow{\bigtriangledown} {\theta_j^T}^t - \sin(\theta_j^T) H\theta_j^T = HC_j^T$, with

$$
\begin{aligned}
\overrightarrow{\bigtriangledown} C_j^T = & \frac{2\eta - p_{t_{j+1}} - p_{t_{j+2}}}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|} - \frac{< p_{t_{j+1}} - \eta | p_{t_{j+2}} - \eta > (\eta - p_{t_{j+2}})}{|p_{t_{j+2}} - \eta|^3 |p_{t_{j+1}} - \eta|} \\
& - \frac{< p_{t_{j+1}} - \eta | p_{t_{j+2}} - \eta > (\eta - p_{t_{j+1}})}{|p_{t_{j+1}} - \eta|^3 |p_{t_{j+2}} - \eta|}
\end{aligned}
\tag{13.2}
$$

and

$$
\begin{aligned}
HC_j^T = & \frac{2I_3}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|} - \frac{(2\eta - p_{t_{j+1}} - p_{t_{j+2}}) \cdot (\eta - p_{t_{j+1}})^t}{|p_{t_{j+2}} - \eta| \cdot |p_{t_{j+1}} - \eta|^3} \\
& - \frac{(2\eta - p_{t_{j+1}} - p_{t_{j+2}}) \cdot (\eta - p_{t_{j+2}})^t}{|p_{t_{j+2}} - \eta|^3 \cdot |p_{t_{j+1}} - \eta|} - \frac{I_3 < p_{t_{j+1}} - \eta | p_{t_{j+2}} - \eta >}{|p_{t_{j+2}} - \eta|^3 |p_{t_{j+1}} - \eta|} \\
& - \frac{(\eta - p_{t_{j+2}}) \cdot (2\eta - p_{t_{j+1}} - p_{t_{j+2}})^t}{|p_{t_{j+2}} - \eta|^3 |p_{t_{j+1}} - \eta|} + \frac{< p_{t_{j+1}} - \eta | p_{t_{j+2}} - \eta > (\eta - p_{t_{j+2}}) \cdot (\eta - p_{t_{j+1}})^t}{|p_{t_{j+2}} - \eta|^3 |p_{t_{j+1}} - \eta|^3} \\
& + \frac{3 < p_{t_{j+1}} - \eta | p_{t_{j+2}} - \eta > (\eta - p_{t_{j+2}}) \cdot (\eta - p_{t_{j+2}})^t}{|p_{t_{j+2}} - \eta|^5 |p_{t_{j+1}} - \eta|} - \frac{I_3 < p_{t_{j+1}} - \eta | p_{t_{j+2}} - \eta >}{|p_{t_{j+1}} - \eta|^3 |p_{t_{j+2}} - \eta|} \\
& - \frac{(\eta - p_{t_{j+1}}) \cdot (2\eta - p_{t_{j+1}} - p_{t_{j+2}})^t}{|p_{t_{j+1}} - \eta|^3 |p_{t_{j+2}} - \eta|} + \frac{< p_{t_{j+1}} - \eta | p_{t_{j+2}} - \eta > (\eta - p_{t_{j+1}}) \cdot (\eta - p_{t_{j+2}})^t}{|p_{t_{j+1}} - \eta|^3 |p_{t_{j+2}} - \eta|^3} \\
& + \frac{3 < p_{t_{j+1}} - \eta | p_{t_{j+2}} - \eta > (\eta - p_{t_{j+1}}) \cdot (\eta - p_{t_{j+1}})^t}{|p_{t_{j+1}} - \eta|^5 |p_{t_{j+2}} - \eta|}
\end{aligned}
\tag{13.3}
$$

Since $(p_{t_k} - \eta)^t \cdot n_T = 0 \ \ \forall k, \forall \eta \in Support(T)$, we have that $\overrightarrow{\bigtriangledown}{\theta_j^T}^t \cdot n_T = 0$ and we obtain:

$$\begin{aligned} {n_T}^t \cdot H\theta_j^T \cdot n_T =& \frac{-{n_T}^t \cdot HC_j^T \cdot n_T}{\sin(\theta_j^T)} \\ =& \frac{\cos(\theta_j^T)}{\sin(\theta_j^T)|p_{t_{j+2}} - \eta|^2} + \frac{\cos(\theta_j^T)}{\sin(\theta_j^T)|p_{t_{j+1}} - \eta|^2} - \frac{2}{\sin(\theta_j^T)|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|} \end{aligned} \tag{13.4}$$

We now focus on the expression of $|N_j^T|^2 {n_T}^t \cdot H\theta_j^T \cdot n_T - \theta_j^T |p_{t_{j+2}} - p_{t_{j+1}}|^2$:

$$\begin{aligned} &|N_j^T|^2 {n_T}^t \cdot H\theta_j^T \cdot n_T - \theta_j^T |p_{t_{j+2}} - p_{t_{j+1}}|^2 = \\ &\frac{|N_j^T|^2 \sin(\theta_j^T)(\cos(\theta_j^T)(|p_{t_{j+2}} - \eta|^2 + |p_{t_{j+2}} - \eta|^2) - 2|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)}{\sin(\theta_j^T)^2 |p_{t_{j+2}} - \eta|^2 |p_{t_{j+1}} - \eta|^2} \\ &- \theta_j^T |p_{t_{j+2}} - p_{t_{j+1}}|^2 \\ =& \sin(\theta_j^T)(\cos(\theta_j^T)(|p_{t_{j+2}} - \eta|^2 + |p_{t_{j+2}} - \eta|^2) - 2|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|) \\ &- \theta_j^T |p_{t_{j+2}} - p_{t_{j+1}}|^2 \end{aligned}$$

Since $|p_{t_{j+2}} - p_{t_{j+1}}|^2 = |p_{t_{j+2}} - \eta|^2 + |p_{t_{j+1}} - \eta|^2 - 2\cos(\theta_j^T)|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|$,

$$\begin{aligned} &|N_j^T|^2 {n_T}^t \cdot H\theta_j^T \cdot n_T - \theta_j^T |p_{t_{j+2}} - p_{t_{j+1}}|^2 \\ &= \sin(\theta_j^T)\cos(\theta_j^T)(|p_{t_{j+2}} - p_{t_{j+1}}|^2 + 2\cos(\theta_j^T)|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta| - 2|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|) \\ &- \theta_j^T |p_{t_{j+2}} - p_{t_{j+1}}|^2 \\ &= (\sin(\theta_j^T)\cos(\theta_j^T) - \theta_j^T)|p_{t_{j+2}} - p_{t_{j+1}}|^2 + 2\cos(\theta_j^T)(\cos(\theta_j^T) - 1)|N_j^T| \end{aligned}$$

This expression is an equivalent of ${\theta_j^T}^3$ around 0, which proves that the problematic term under the second sum in the expression of $N_i^T(\eta + \epsilon n_T)^t \cdot m^T(\eta + \epsilon n_T)$ tends to 0, and can be neglected in the calculus of $\overrightarrow{\bigtriangledown} w_i^T$ in the special case of $\eta \in Support(T), \notin T$.

Finally, $\forall \eta \in Support(T), \notin T$:

$$\begin{aligned} -2|T|\overrightarrow{\bigtriangledown} w_i^T =& \sum_j \frac{e_2(\theta_j^T)(p_{t_{i+2}} - p_{t_{i+1}})^t \cdot (p_{t_{j+2}} - p_{t_{j+1}})}{2|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|} n_T \\ &+ \sum_j \frac{e_1(\theta_j^T)|p_{t_{j+2}} - p_{t_{j+1}}|^2 {N_i^T}^t \cdot N_j^T}{4(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} n_T \\ &+ \sum_j \frac{\cos(\theta_j^T) e_3(\theta_j^T) {N_i^T}^t \cdot N_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^2} n_T \end{aligned}$$

with e_1, e_2, and e_3 being functions well defined on $]0, \pi[$ and that admit controllable Taylor expansion around 0.

13.2 Appendix B: Expression of the Hessian in 3D, in the case of alignment with the triangle T

We have seen that if $\eta \in Support(T), \notin T$, the gradient of the unnormalized weight with regard to T is given by $\overrightarrow{\bigtriangledown} w_i^T(\eta) = dw_i^T(\eta) n_T$, with $dw_i^T(\eta)$ a scalar function whose form was presented previously. In order to get the Hessian of the unnormalized weight with regards to T, we need to derivate dw_i^T since $Hw_i^T(\eta) = n_T \cdot \overrightarrow{\bigtriangledown} dw_i^T{}^t$.

Expression of $\overrightarrow{\bigtriangledown} dw_i^T$ By derivating the expression of $\frac{\partial(w_i^T(\eta+\epsilon n_T))}{\partial \epsilon}_{|\epsilon \to 0}$ that we obtained previously, we have that

$$
\begin{aligned}
-2|T|\overrightarrow{\bigtriangledown} dw_i^T = & \sum_j \frac{((p_{t_{i+2}} - p_{t_{i+1}})^t \cdot (p_{t_{j+2}} - p_{t_{j+1}}))\overrightarrow{\bigtriangledown}\theta_j^T}{2|N_j^T|} \\
& - \sum_j \frac{\theta_j^T((p_{t_{i+2}} - p_{t_{i+1}})^t \cdot (p_{t_{j+2}} - p_{t_{j+1}}))JN_j^{T^t} \cdot N_j^T}{2|N_j^T|^3} \\
& - \sum_j \frac{\sin(\theta_j^T)^2 |p_{t_{j+2}} - p_{t_{j+1}}|^2 (N_i^{T^t} \cdot N_j^T)\overrightarrow{\bigtriangledown}\theta_j^T}{2|N_j^T|^3} \\
& + \sum_j \frac{(\sin(\theta_j^T)\cos(\theta_j^T) - \theta_j^T)|p_{t_{j+2}} - p_{t_{j+1}}|^2 (JN_j^{T^t} \cdot N_i^T + JN_i^{T^t} \cdot N_j^T)}{4|N_j^T|^3} \\
& - \sum_j \frac{3(\sin(\theta_j^T)\cos(\theta_j^T) - \theta_j^T)|p_{t_{j+2}} - p_{t_{j+1}}|^2 (N_i^{T^t} \cdot N_j^T)JN_j^{T^t} \cdot N_j^T}{4|N_j^T|^5} \\
& + \sum_j \frac{\sin(\theta_j^T)(1 - 2\cos(\theta_j^T))(N_i^{T^t} \cdot N_j^T)\overrightarrow{\bigtriangledown}\theta_j^T}{2|N_j^T|^2} \\
& + \sum_j \frac{\cos(\theta_j^T)(\cos(\theta_j^T) - 1)(JN_j^{T^t} \cdot N_i^T + JN_i^{T^t} \cdot N_j^T)}{2|N_j^T|^2} \\
& - \sum_j \frac{\cos(\theta_j^T)(\cos(\theta_j^T) - 1)(N_i^{T^t} \cdot N_j^T)JN_j^{T^t} \cdot N_j^T}{|N_j^T|^4}
\end{aligned}
$$

By replacing $\overrightarrow{\bigtriangledown}\theta_j^T$ by the expression found at the Eq. 10.23, we have

$$
\begin{aligned}
-2|T|\overrightarrow{\bigtriangledown} dw_i^T = &-\sum_j \frac{\sin(\theta_j^T)^2((p_{t_{i+2}}-p_{t_{i+1}})^t\cdot(p_{t_{j+2}}-p_{t_{j+1}}))(2\eta-p_{t_{j+2}}-p_{t_{j+1}})}{2|N_j^T|^2}\\
&+\sum_j \frac{\cos(\theta_j^T)\sin(\theta_j^T)((p_{t_{i+2}}-p_{t_{i+1}})^t\cdot(p_{t_{j+2}}-p_{t_{j+1}}))JN_j^{T^t}\cdot N_j^T}{2|N_j^T|^3}\\
&-\sum_j \frac{\theta_j^T((p_{t_{i+2}}-p_{t_{i+1}})^t\cdot(p_{t_{j+2}}-p_{t_{j+1}}))JN_j^{T^t}\cdot N_j^T}{2|N_j^T|^3}\\
&+\sum_j \frac{\sin(\theta_j^T)^4|p_{t_{j+2}}-p_{t_{j+1}}|^2(N_i^{T^t}\cdot N_j^T)(2\eta-p_{t_{j+2}}-p_{t_{j+1}})}{2|N_j^T|^4}\\
&+\sum_j \frac{(\sin(\theta_j^T)\cos(\theta_j^T)-\theta_j^T)|p_{t_{j+2}}-p_{t_{j+1}}|^2(JN_j^{T^t}\cdot N_i^T+JN_i^{T^t}\cdot N_j^T)}{4|N_j^T|^3}\\
&-\sum_j \frac{\cos(\theta_j^T)\sin(\theta_j^T)^3|p_{t_{j+2}}-p_{t_{j+1}}|^2(N_i^{T^t}\cdot N_j^T)JN_j^{T^t}\cdot N_j^T}{2|N_j^T|^5}\\
&-\sum_j \frac{3(\sin(\theta_j^T)\cos(\theta_j^T)-\theta_j^T)|p_{t_{j+2}}-p_{t_{j+1}}|^2(N_i^{T^t}\cdot N_j^T)JN_j^{T^t}\cdot N_j^T}{4|N_j^T|^5}\\
&+\sum_j \frac{\cos(\theta_j^T)(\cos(\theta_j^T)-1)(JN_j^{T^t}\cdot N_i^T+JN_i^{T^t}\cdot N_j^T)}{2|N_j^T|^2}\\
&+\sum_j \frac{\sin(\theta_j^T)^3(1-2\cos(\theta_j^T))(N_i^{T^t}\cdot N_j^T)(2\eta-p_{t_{j+2}}-p_{t_{j+1}})}{2|N_j^T|^3}\\
&+\sum_j \frac{\cos(\theta_j^T)\sin(\theta_j^T)^2(1-2\cos(\theta_j^T))(N_i^{T^t}\cdot N_j^T)JN_j^{T^t}\cdot N_j^T}{2|N_j^T|^4}\\
&-\sum_j \frac{\cos(\theta_j^T)(\cos(\theta_j^T)-1)(N_i^{T^t}\cdot N_j^T)JN_j^{T^t}\cdot N_j^T}{|N_j^T|^4}
\end{aligned}
$$

$$
\begin{aligned}
-2|T|\overrightarrow{\bigtriangledown} dw_i^T = &-\sum_j \frac{((p_{t_{i+2}} - p_{t_{i+1}})^t \cdot (p_{t_{j+2}} - p_{t_{j+1}}))(2\eta - p_{t_{j+2}} - p_{t_{j+1}})}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^2} \\
&+\sum_j \frac{e_1(\theta_j^T)((p_{t_{i+2}} - p_{t_{i+1}})^t \cdot (p_{t_{j+2}} - p_{t_{j+1}}))JN_j^{T^t} \cdot N_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} \\
&+\sum_j \frac{|p_{t_{j+2}} - p_{t_{j+1}}|^2(N_i^{T^t} \cdot N_j^T)(2\eta - p_{t_{j+2}} - p_{t_{j+1}})}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^4} \\
&+\sum_j \frac{e_1(\theta_j^T)|p_{t_{j+2}} - p_{t_{j+1}}|^2(JN_j^{T^t} \cdot N_i^T + JN_i^{T^t} \cdot N_j^T)}{4(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} \\
&-\sum_j \frac{e_4(\theta_j^T)|p_{t_{j+2}} - p_{t_{j+1}}|^2(N_i^{T^t} \cdot N_j^T)JN_j^{T^t} \cdot N_j^T}{4(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^5} \\
&+\sum_j \frac{\cos(\theta_j^T)e_3(\theta_j^T)(JN_j^{T^t} \cdot N_i^T + JN_i^{T^t} \cdot N_j^T)}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^2} \\
&+\sum_j \frac{(1 - 2\cos(\theta_j^T))(N_i^{T^t} \cdot N_j^T)(2\eta - p_{t_{j+2}} - p_{t_{j+1}})}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^3} \\
&+\sum_j \frac{e_5(\theta_j^T)(N_i^{T^t} \cdot N_j^T)JN_j^{T^t} \cdot N_j^T}{2(|p_{t_{j+2}} - \eta||p_{t_{j+1}} - \eta|)^4}
\end{aligned}
$$

where

$$e_1(x) = \frac{\cos(x)\sin(x) - x}{\sin(x)^3} \tag{13.5}$$

$$e_4(x) = \frac{2\cos(x)\sin(x)^3 + 3(\sin(x)\cos(x) - x)}{\sin(x)^5} \tag{13.6}$$

$$e_3(x) = \frac{\cos(x) - 1}{\sin(x)^2} \tag{13.7}$$

$$e_5(x) = \frac{\cos(x)\sin(x)^2(1 - 2\cos(x)) - 2\cos(x)^2 + 2\cos(x)}{\sin(x)^4} \tag{13.8}$$

are functions well defined on $]0, \pi[$ and that admit controllable Taylor expansions around 0.

13.3 Appendix C: Details of Taylor expansion of $w_i^E(\eta + \epsilon n_E)$ in the 2D case

Expression of $\frac{\partial(w_i^E(\eta+\epsilon n_E))}{\partial\epsilon}_{|\epsilon\to 0}$ To obtain a closed-form expression of $\frac{\partial(w_i^E(\eta+\epsilon n_E))}{\partial\epsilon}_{|\epsilon\to 0}$, we derive the Taylor expansion of $w_i^E(\eta + \epsilon n_E)$ with regards to ϵ.

$$w_i^E(\eta + \epsilon n_E) = \frac{m^E(\eta + \epsilon n_E)^t \cdot N_{i+1}^E(\eta + \epsilon n_E)}{(p_{e_i} - \eta - \epsilon n_E)^t \cdot N_{i+1}^E(\eta + \epsilon n_E)} \tag{13.9}$$

The denominator of the latter fraction is equal to:

$$\begin{aligned}
&(p_{e_i} - \eta - \epsilon n_E)^t \cdot N_{i+1}^E(\eta + \epsilon n_E) = \\
&(p_{e_i} - \eta - \epsilon n_E)^t \cdot R_{\frac{\pi}{2}} \cdot (p_{e_{i+1}} - \eta - \epsilon n_E) \cdot (-1)^i \\
=&(p_{e_i} - \eta)^t \cdot R_{\frac{\pi}{2}} \cdot (p_{e_{i+1}} - \eta) \cdot (-1)^i - \epsilon n_E^t \cdot R_{\frac{\pi}{2}} \cdot (p_{e_{i+1}} - \eta) \cdot (-1)^i \\
&- \epsilon(p_{e_i} - \eta)^t \cdot R_{\frac{\pi}{2}} \cdot n_E \cdot (-1)^i + \epsilon^2 n_E^t R_{\frac{\pi}{2}} n_E \cdot (-1)^i \\
=&\epsilon n_E^t \cdot R_{\frac{\pi}{2}} \cdot (p_{e_i} - p_{e_{i+1}}) \cdot (-1)^i \\
=&-\epsilon|E|
\end{aligned}$$

The numerator is equal to:

$$m^E(\eta + \epsilon n_E)^t \cdot N_{i+1}^E(\eta + \epsilon n_E) = \sum_j \frac{N_j^E(\eta + \epsilon n_E)^t \cdot N_{i+1}^E(\eta + \epsilon n_E)}{|N_j^E(\eta + \epsilon n_E)|}$$

We develop $N_j^E(\eta + \epsilon n_E)^t \cdot N_{i+1}^E(\eta + \epsilon n_E)$ to the second order:

$$\begin{aligned}
&N_j^E(\eta + \epsilon n_E)^t \cdot N_{i+1}^E(\eta + \epsilon n_E) = \\
&(N_j^E(\eta) + \epsilon(-1)^{j+1} R_{\frac{\pi}{2}} \cdot n_E)^t \cdot (N_{i+1}^E(\eta) + \epsilon(-1)^i R_{\frac{\pi}{2}} \cdot n_E) \\
=&N_j^E(\eta)^t \cdot N_{i+1}(\eta) + \epsilon(-1)^{j+1} n_E^t \cdot N_{i+1}^E(\eta) \\
&+ \epsilon(-1)^i N_j^E(\eta)^t \cdot R_{\frac{\pi}{2}} \cdot n_E + \epsilon^2(-1)^{i+j+1}(R_{\frac{\pi}{2}} \cdot n_E)^t \cdot (R_{\frac{\pi}{2}} \cdot n_E) \\
=&N_j^E(\eta)^t \cdot N_{i+1}(\eta) + \epsilon^2(-1)^{i+j+1}
\end{aligned}$$

We develop $\frac{1}{|N_j^E(\eta+\epsilon n_E)|}$ to the second order:

$$\begin{aligned}\frac{1}{|N_j^E(\eta+\epsilon n_E)|} =& \frac{1}{|N_j^E|} - \epsilon\frac{(JN_j^{E^t}\cdot N_j^E)^t\cdot n_E}{|N_j^E|^3}\\ &- \frac{\epsilon^2}{2}\frac{n_E^t\cdot(JN_j^{E^t}\cdot JN_j^E)\cdot n_E}{|N_j^E|^3}\\ &+ 3\frac{\epsilon^2}{2}\frac{n_E^t\cdot(JN_j^{E^t}\cdot N_j^E\cdot N_j^{E^t}\cdot JN_j^E)\cdot n_E}{|N_j^E|^5}\\ =& \frac{1}{|N_j^E|} - \epsilon^2\frac{1}{2|N_j^E|^3}\end{aligned}$$

We obtain the Taylor expansion of $m^E(\eta+\epsilon n_E)^t\cdot N_{i+1}^E(\eta+\epsilon n_E)$ as:

$$\begin{aligned}&m^E(\eta+\epsilon n_E)^t\cdot N_{i+1}^E(\eta+\epsilon n_E)\\ &\quad=\sum_j (N_j^E(\eta)^t\cdot N_{i+1}(\eta)+\epsilon^2(-1)^{i+j+1})(\frac{1}{|N_j^E|}-\epsilon^2\frac{1}{2|N_j^E|^3})+o(\epsilon^2)\\ &\quad=m^E(\eta)^t\cdot N_{i+1}^E(\eta)-\epsilon^2\sum_j\frac{N_j^E(\eta)^t\cdot N_{i+1}^E(\eta)}{2|N_j^E(\eta)|^3}+\frac{(-1)^{i+j}}{|N_j^E(\eta)|}+o(\epsilon^2)\\ &\quad=-\epsilon^2\sum_j\frac{N_j^E(\eta)^t\cdot N_{i+1}^E(\eta)}{2|N_j^E(\eta)|^3}+\frac{(-1)^{i+j}}{|N_j^E(\eta)|}+o(\epsilon^2)\end{aligned}$$

We obtain that, for all $\eta\in(e_0e_1),\notin[e_0e_1]$:

$$\overrightarrow{\bigtriangledown} w_i^E(\eta)=(\sum_j\frac{N_j^E(\eta)^t\cdot N_{i+1}^E(\eta)}{2|E||N_j^E(\eta)|^3}+\frac{(-1)^{i+j}}{|E||N_j^E(\eta)|})n_E \tag{13.10}$$

13.4 Appendix D: Proof of Eq. 11.2: (Green biharmonic theorem)

From Stokes' theorem, we know that $\int_{\xi\in D} div_\xi(\overrightarrow{F})(\xi)d\rho_\xi = \int_{\xi\in\partial D} \overrightarrow{F}(\xi)\cdot n_\xi d\sigma_\xi$ (where $d\rho_\xi$ is the volumic element, $d\sigma_\xi$ is the surfacic element, and n_ξ is the outward unit normal on the boundary of the domain).
Considering the following equations (with $\lambda \triangleq \lambda(\xi), g \triangleq g(\xi,\eta)$ and all derivatives being expressed w.r.t. ξ):

$$\begin{cases} div(\overrightarrow{\bigtriangledown}(\triangle\ \lambda)g) = \triangle^2\ (\lambda)g + \overrightarrow{\bigtriangledown}(\triangle\ (\lambda))\cdot\overrightarrow{\bigtriangledown}(g) & (a)\\ div(\triangle\ \lambda\overrightarrow{\bigtriangledown}g) = \overrightarrow{\bigtriangledown}(\triangle\ (\lambda))\cdot\overrightarrow{\bigtriangledown}(g) + \triangle\ (\lambda)\ \triangle\ (g) & (b)\\ div(\overrightarrow{\bigtriangledown}\lambda\ \triangle\ g) = \triangle\ (\lambda)\ \triangle\ (g) + \overrightarrow{\bigtriangledown}(\lambda)\cdot\overrightarrow{\bigtriangledown}(\triangle\ (g)) & (c)\\ div(\lambda\overrightarrow{\bigtriangledown}(\triangle\ g)) = \overrightarrow{\bigtriangledown}(\lambda)\cdot\overrightarrow{\bigtriangledown}(\triangle\ (g)) + \lambda\ \triangle^2\ (g) & (d)\end{cases}$$

These equations can be obtained with the following observation: for any ***n*-dimensional** function $\overrightarrow{U}$, for any **scalar** function v, $div(\overrightarrow{U}v) = \sum_{x_i}\partial_{x_i}(U_i v) = \sum_{x_i}\partial_{x_i}(U_i)v + \sum_{x_i} U_i\partial_{x_i}(v) = div(\overrightarrow{U})v + \overrightarrow{U}\cdot\overrightarrow{\bigtriangledown}v$.

By writing $(d)-(c)+(b)-(a)$, we have that
$\lambda\ \triangle^2\ (g) - \triangle^2\ (\lambda)g = div(\lambda\overrightarrow{\bigtriangledown}(\triangle\ g)) - div(\overrightarrow{\bigtriangledown}\lambda\ \triangle\ g) + div(\triangle\ \lambda\overrightarrow{\bigtriangledown}g) - div(\overrightarrow{\bigtriangledown}(\triangle\ \lambda)g)$
Now, making the remark that λ is writen to be biharmonic ($\triangle^2\ (\lambda) = 0$), and that g is the biharmonic Green function ($\triangle\ (g) = G$ and $\triangle^2\ (g)$ is the Dirac operator δ_0), we have that

$$\begin{aligned}\lambda(\eta) &= \int_{\xi\in D}\lambda(\xi)\delta_0(||\xi-\eta||)d\rho_\xi\\ &= \int_{\xi\in D} div_\xi(\lambda(\xi)\overrightarrow{\bigtriangledown}_\xi(G)(\xi,\eta))d\rho_\xi\\ &\quad - \int_{\xi\in D} div_\xi(\overrightarrow{\bigtriangledown}_\xi\lambda(\xi)G(\xi,\eta))d\rho_\xi\\ &+ \int_{\xi\in D} div_\xi(\triangle_\xi\ \lambda(\xi)\overrightarrow{\bigtriangledown}_\xi g(\xi,\eta))d\rho_\xi\\ &- \int_{\xi\in D} div_\xi(\overrightarrow{\bigtriangledown}_\xi(\triangle_\xi\ \lambda)(\xi)g(\xi,\eta))d\rho_\xi\end{aligned}$$

Applying Stokes' theorem to the right hand term of this last equation completes the proof. Note, that the writing of this proof is correct for any scalar function λ. Writing it per component extends it directly to any multi-dimensional biharmonic function.