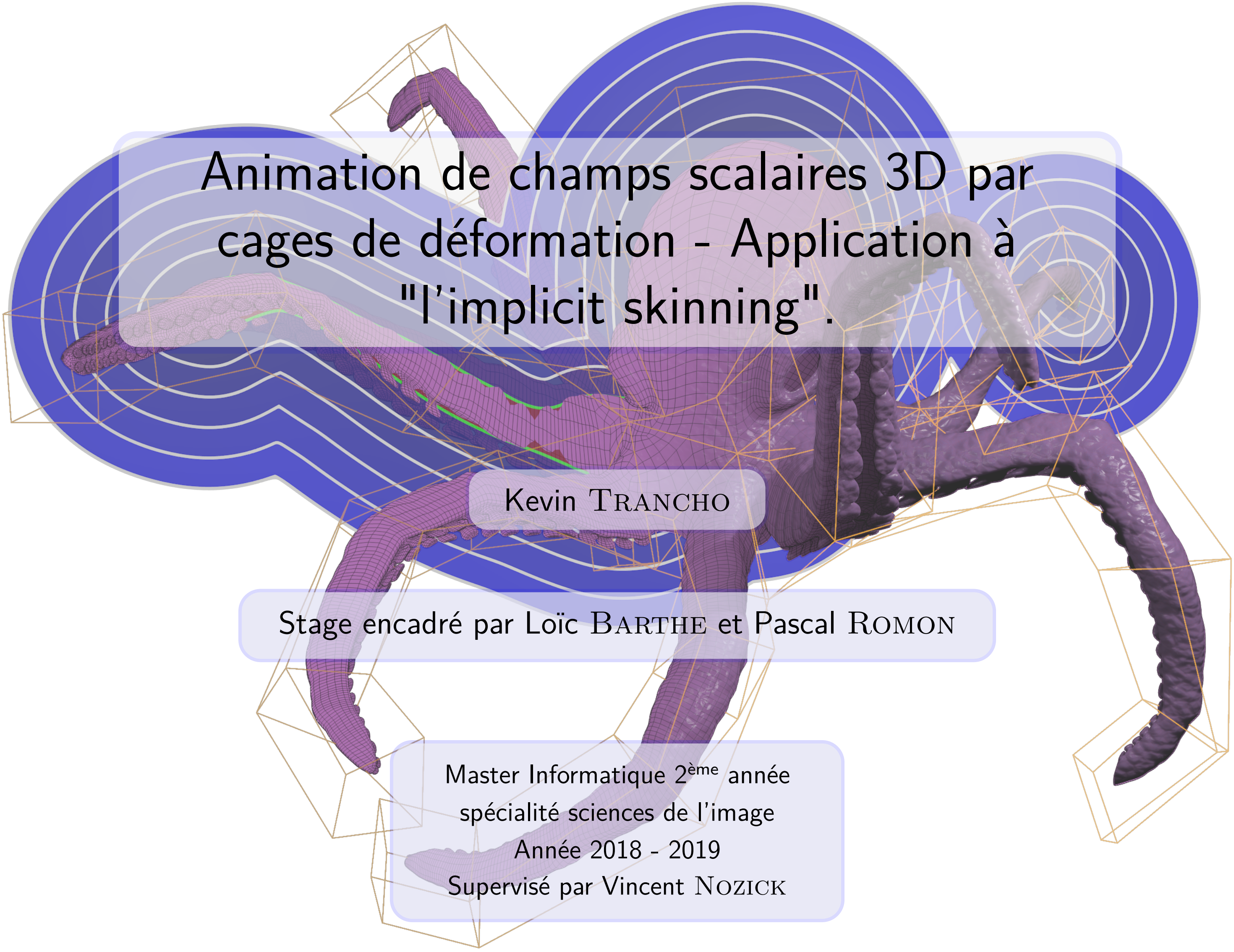


Animation de champs scalaires 3D par cages de déformation - Application à "l'implicit skinning".

Kevin Trancho

Stage encadré par Loïc Barthe et Pascal Romon

Master Informatique 2ème année
spécialité sciences de l'image
Année 2018 - 2019
Supervisé par Vincent Nozick

Table des matières

Table des figures

Remerciements

Je tiens tout d'abord à remercier Loïc Barthe aussi bien pour m'avoir donné la possibilité de travailler sur un sujet stimulant et intéressant que pour m'avoir fait confiance et laissé une certaine autonomie pour mener mes travaux tout en étant présent pour échanger et m'apporter son aide pour avancer dans mon stage. J'aimerais de même remercier Pascal Romon pour m'avoir soutenu et accompagné dans mon projet de stage ainsi que pour s'être montré disponible pour échanger et suivre mes travaux.

Je souhaite aussi attribuer un remerciement particulier à Florian Canezin à la fois pour les échanges intéressant que nous avons pu avoir sur mon projet que l'aide et les explications techniques qu'il m'a apporté sur Radium-Engine et l'Implicit skinning. De même, je remercie Paul Kry pour les échanges intéressants que nous avons eu sur mes travaux, ainsi que Nicolas Mellado et aussi Brian Wyvill pour les avoir suivi.

J'aimerais aussi remercier les membres de l'équipe STORM que j'ai pu rencontrer durant mon stage à l'IRIT, c'est-à-dire Anahid, Chems-Eddine, Clément, David, Filippo, François, Hugo, Kévin, Mathias, Olivier, Pierre, Robin, Solange et Thibault.

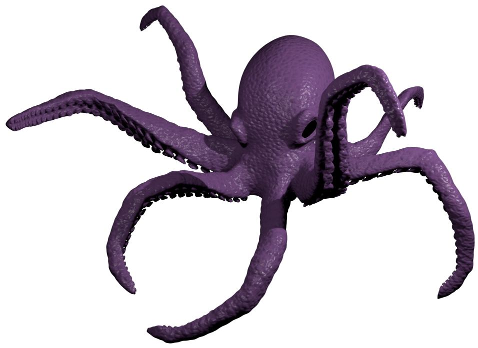
Je souhaite attribuer mes remerciements personnels à ma famille, en particulier ma grand-mère Josette qui m'a hébergé chez elle au début de mon stage, ainsi qu'à mes parents Ghislain et Valérie pour leur soutien dans tous les domaines pour effectuer au mieux mon stage à Toulouse.

Première partie
Introduction

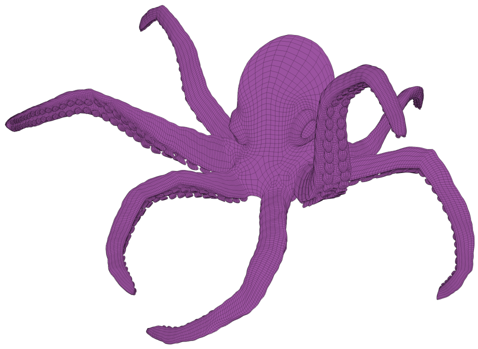
1 Déformations géométriques pour la synthèse d'image

1.1 Maillages et représentation d'entités

Nos écrans nous proposent des résultats toujours plus spectaculaires et réalistes, que ce soit dans les jeux vidéos, les effets spéciaux au cinéma, les films d'animation ou encore la réalité virtuelle. Il est donc naturel de se demander quelles sont les techniques qui produisent ces résultats et comment aller plus loin. Bien souvent dans l'industrie nous représentons une forme ou une entité à l'aide de triangles, quadrilatères et autres polygones dont l'ensemble, que l'on nomme maillage, forme une représentation expressive à l'œil humain comme illustré en figure 1.



Rendu final



Rendu de la géométrie

FIGURE 1 – **Exemple de rendu d'un maillage sous Blender représentant une pieuvre**

1.2 Animation et skinning

FIGURE 2 – **Exemple d'animation basée squelette**
En déformant un squelette d'animation (en vert), un animateur donne vie à son personnage. Le maillage se déforme ensuite selon l'animation donnée par le squelette.

Un simple maillage en tant que tel possède une sémantique, mais bien souvent, nous sommes habitués à voir les entités s'animer. Pour produire ceci, il est proposé de déformer les maillages pour leur donner vie. Nos entités sont souvent équipées d'un squelette d'animation qui mime le comportement d'un squelette anatomique tel qu'on le connaît. C'est un outil de déformation utilisé par les studios d'animation qui permet de proposer des transformations rigides de notre forme ce que l'on nomme le *skinning* (illustré en figure 2).

L'outil de déformation que nous étudierons est celui des cages de déformations. Leur intérêt est de proposer des déformations moins rigides et plus libres pour déformer un maillage. Ce qui est intéressant pour animer des invertébrés (figure 3) et d'autres entités.

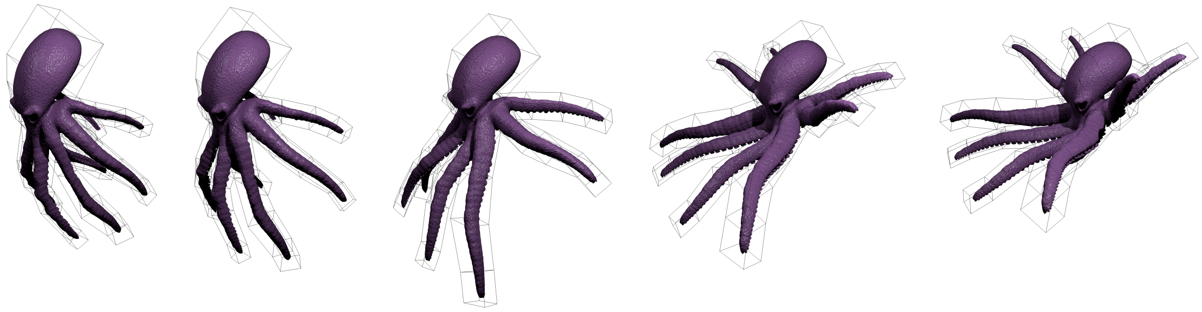


Figure 3 – **Exemple d'animation basée cage**

1.3 Implicit skinning

L'Implicit skinning [Vaillant et al., 2013] est une méthode de déformation récente basée squelette. Elle permet de proposer la correction du maillage à l'aide de surfaces implicites [Gourmel et al., 2013] principalement aux régions coudées et aux plis, là où les méthodes actuellement utilisées dans l'industrie comme le *Linear Bending Skinning* ou le *Dual Quaternion Skinning* demandent aux infographistes un travail plus fastidieux pour obtenir une déformation plausible.

Implicit skinning — Dual Quaternion Skinning — Linear Bending Skinning

Figure 4 – **Illustration des méthodes de skinning sur le modèle *Dana***

L'illustration en figure 4 est réalisée depuis la démonstration proposée par l'équipe STORM. De même depuis 2018, un plugin Maya est accessible pour une future utilisation de l'Implicit skinning en production.

2 L'IRIT et l'équipe STORM

2.1 Contexte

Dans le cadre de ma dernière année en master d'informatique je suis tenu d'effectuer un stage en entreprise, j'ai donc profité de cette opportunité pour effectuer un stage en laboratoire de recherche à l'**IRIT** (*Institut de Recherche en Informatique de Toulouse*). Passionné par la modélisation 3D, la déformation de formes et l'aspect géométrique de l'infographie, j'ai rejoint l'équipe **STORM** (*Structural Models and Tools in Computer Graphics*).

J'y ai été encadré par **Loïc Barthe**, professeur à l'Université Paul Sabatier et chercheur en informatique graphique à l'**IRIT**. Ses travaux s'intéressent principalement à l'aspect géométrique de la modélisation 3D, l'animation et l'étude de modèles tels que les maillages ou les surfaces implicites ainsi que leur déformation.

J'ai aussi l'occasion de partager mon intérêt pour l'Implicit skinning avec **Pascal Romon**, maître de conférences à l'Université Paris-Est Marne-la-Vallée et chercheur en mathématiques au **LAMA** (*Laboratoire d'Analyse et de Mathématiques Appliquées*), qui a souhaité m'accompagner sur mon projet de stage. Il enseigne la géométrie différentielle discrète, plus particulièrement les flots de courbure pour la déformation de maillages et s'intéresse aux aspects mathématiques de la géométrie.

2.2 L'IRIT

L'IRIT se situe à Toulouse sur le campus scientifique de l'Université Paul Sabatier. Initialement fondé en 1990, l'IRIT, une **UMR** (*Unité Mixte de Recherche*) entre le CNRS, L'Université Paul Sabatier et l'Institut Polytechnique de Toulouse, est issue de la fusion du **LSI** (*Laboratoire de Langages et Systèmes Informatiques*) et du **CERFIA** (*Cybernétique des Entreprises, Reconnaissance de Formes et Intelligence Artificielle*). Aujourd'hui, l'IRIT c'est 275 chercheurs/enseignants-chercheurs, 200 doctorants ainsi que 292 post-doctorants/contractuels qui y travaillent et s'y divisent en 24 équipes de recherche. Pour donner une idée des différentes thématiques de recherches abordées à l'IRIT, je citerai ses différents départements :

- **Signaux, images** : s'intéresse principalement aux domaines qui vont du traitement de la vidéo à la reconnaissance de la parole, ceci à l'aide d'outils tels que l'optimisation, l'apprentissage automatique ou le traitement statistique pour des applications dans l'imagerie satellitaire ou médicale. Mais aussi à la synthèse de l'image aussi bien sur l'aspect rendu que géométrique.
- **Gestion des données** : étudie l'information en général dans le but de proposer des modèles et algorithmes qui traitent des données de l'ordre du texte, de faits ou de données quelconques.
- **Intelligence collective, interaction** : s'inscrit principalement dans les domaines qui touchent à l'interaction Homme-Machine, aussi bien pour la perception ou le comportement côté humain que pour y proposer des modèles et technologies côté machine, mais aussi aux systèmes multi-agents tels que l'intelligence artificielle distribuée ayant par exemple pour application une *ville intelligente*.
- **Intelligence artificielle** : s'intéresse au raisonnement et à la prise de décision automatisée, ceci dans le but d'aider l'humain. Cela intervient dans des domaines tels que la théorie des jeux, les réseaux sociaux, les protocoles de sécurité ou encore le discours.
- **Calcul intensif, simulation, optimisation** : étudie le développement d'algorithmes et leur implémentation pour la résolution en parallèle de problèmes mathématiques. Ceci pour des applications qui vont de la vision par ordinateur à la biologie computationnelle, mais aussi en météorologie ou géophysique.
- **Architecture, systèmes, réseaux** : s'intéresse aux protocoles réseaux aussi bien satellites, qu'embarqués ou distribués. Ceci dans un optique temps-réel et en étudie la fiabilité des échanges.
- **Fiabilité des logiciels** : étudie la vérification formelle de logiciels aussi bien par des preuves formelles que l'étude de systèmes critiques embarqués et distribués.

2.3 l'équipe STORM

Je suis affecté au département **Signaux, images**, dans l'équipe STORM, actuellement dirigée par Mathias Paulin depuis 2017. Elle est issue de l'équipe **VORTEX** (*Visual Objects from Reality To Expression*), anciennement crée en 2006 et dirigée par Hervé Luga. L'équipe STORM regroupe à la fois les thématiques qui touchent à l'**apparence** et la **géométrie** en synthèse d'images.

L'équipe s'intéresse à des thématiques variées en terme de rendu, c'est-à-dire qu'elle s'intéresse aussi bien à l'aspect **temps-réel** du rendu dont l'implémentation CPU et GPU [Pajot et al., 2011], que l'aspect **expressif** dans un intérêt artistique par la représentation de la géométrie [Vergne et al., 2011] ou encore l'aspect **perception** et changements de couleurs [Mellado et al., 2017]. Mais elle étudie aussi le rendu basé sur la simulation de lumière ou d'ombres [Forest et al., 2009] et principalement l'utilisation de techniques probabilistes de résolution d'intégrales basées **Monte-Carlo**.

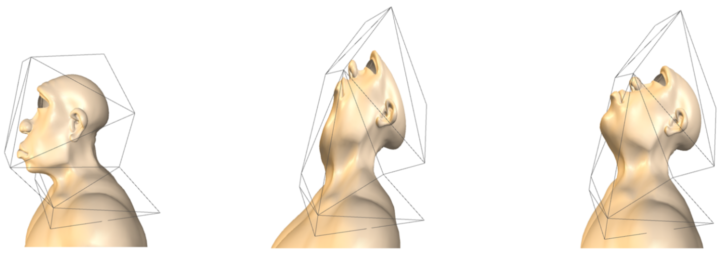
Concernant l'aspect géométrique, les thématiques abordées vont du traitement de nuages de points [Tachella et al., 2019] et reconnaissances de motifs [Biasotti et al., 2018] jusqu'à l'animation de modèles. La thématique grandement étudiée par l'équipe STORM est celle des **surfaces implicites** aussi bien que pour la modélisation de formes [Gourmel et al., 2013] et le rendu efficace [Gourmel et al., 2010] ainsi que leur utilisation en tant qu'outil pour l'animation de maillages depuis un squelette articulé [Vaillant et al., 2013] [Vaillant et al., 2014] [Roussellet et al., 2018].

3 Travail de stage et intérêt

3.1 Déformation par les cages de surfaces implicites

L'Implicit skinning est uniquement basé squelette et ne propose que des déformation au niveau des articulations des modèles animés. Une première amélioration pour proposer des déformations moins rigides et des effets physiques est l'ajout de muscles plausibles [Roussellet et al., 2018]. Pour continuer dans cette lancée, nous souhaiterions laisser la possibilité aux infographistes d'utiliser les cages de déformation en général sur les surfaces implicites et en particulier pour l'Implicit skinning. En effet, bien souvent, un animateur peut souhaiter avoir un plus grand contrôle sur la déformation pour exagérer des effets ou être plus libre sur sa façon de déformer.

De plus, il est possible d'équiper une déformation basée cage d'un système de coordonnées barycentriques qui conditionnera la déformation, ceci pour offrir une déformation qui s'accorde avec la déformation de la cage [Ju et al., 2005] ou avec ses changements d'orientation pour des déformation plus rigides [Lipman et al., 2008] (figure 5).



Maillage et cage d'origine | *Mean Value Coordinates* [Ju et al., 2005] | *Green coordinates* [Lipman et al., 2008]

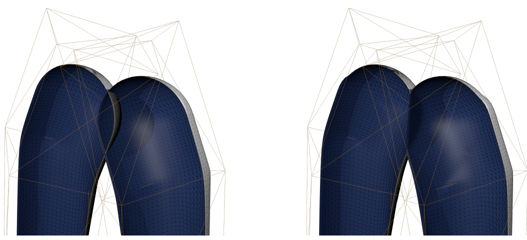
(Images de [Lipman et al., 2008])

FIGURE 5 – **Illustration de déformations par des systèmes de coordonnées barycentriques**

Pour proposer ce type de déformation à l'Implicit skinning, mon travail consiste à déformer une surface implicite à l'aide d'une cage de déformation. Pour obtenir la surface déformée, ceci demande de résoudre un problème de position inverse pour lequel nous avons étudié deux méthodes : une première qui se base sur la résolution directe du problème et une seconde qui se base sur la capture de la déformation pour approximer une solution.

3.2 Correction du maillage et auto-intersections de la surface implicite

Grâce à la détection d'auto-intersections de la surface implicite, mes travaux se sont aussi orientés sur la composition de celle-ci avec elle-même. L'intérêt est de corriger et produire des effets liés aux contacts lors de déformations par les cages lorsqu'un un maillage entre en collision avec lui-même (figure 6).



Maillage déformé | Maillage corrigé

FIGURE 6 – **Vue en coupe de la correction du maillage lors d'une auto-intersection**

Deuxième partie
Implicit skinning

Pour donner au lecteur les prérequis pour comprendre mes travaux, j'expliquerai brièvement comment fonctionnent les déformations basées squelettes. Puis, ce que sont les surfaces implicites et leur composition. Finalement, je donnerai une vue d'ensemble de l'Implicit skinning, ainsi que les améliorations de la méthode qui sont à prendre en compte pour mes travaux.

4 Déformation basée squelette

4.1 Rigging

Un squelette d'animation tel que présenté précédemment est pour l'infographiste un ensemble d'os liés par des liens de parenté ce qui forme un structure qu'il peut transformer en effectuant des rotations des os, le déplacement des extrémités ou encore des changements d'échelle. Du point de vue informatique, c'est un arbre pour lequel chaque nœud définit une transformation locale de d'espace : rotations, translations et homothéties. Pour faire le lien entre le maillage et le squelette d'animation, il est proposé à l'infographiste d'associer pour chaque sommet v des poids $w_{\mathfrak{B}_i}(v) \in [0,1]$ par rapport à chaque articulation $\mathfrak{B}_i$ pour l'indiquer l'influence de chaque os en un sommet. Ceci est illustré en figure 7 pour les 3 articulations du doigt d'une main. Cette étape de définition de la structure pour la déformation est nommée le *rigging*.

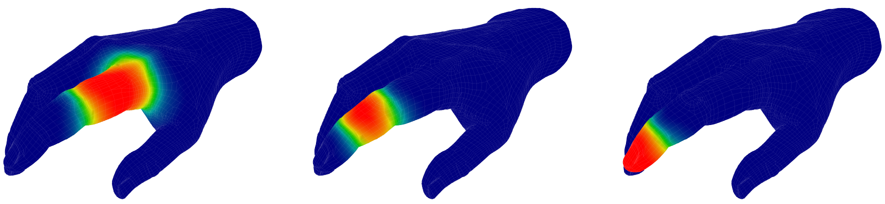


FIGURE 7 – **Rigging d'une main : exemple d'association de poids sommets/articulations** (Influence maximale en rouge et aucune influence en bleu)

Le *rigging* prépare l'étape de déformation du maillage : le *skinning*, dont je présenterai rapidement deux méthodes actuellement utilisées dans l'industrie et leurs principaux défauts. Puis, je donnerai un aperçu de l'Implicit skinning.

4.2 Linear Bending Skinning

Le *Linear Bending Skinning*, initialement introduit par [Magnenat-Thalmann et al., 1989] est principalement utilisé pour sa rapidité de calcul. Dans un premier temps, pour chaque transformation du type rotation, translation ou homothétie on peut associer une matrice de transformation. Ces matrices transformeront un point ou un vecteur depuis ses coordonnées homogènes. Par exemple, un point en 2D est donné par un vecteur $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ et un vecteur en 2D par $\begin{pmatrix} v_x \\ v_y \\ 0 \end{pmatrix}$. Les coordonnées homogènes ont pour but de rendre les vecteurs invariants aux translations et manipuler points et vecteurs par les mêmes transformations, illustrées en figure 8.

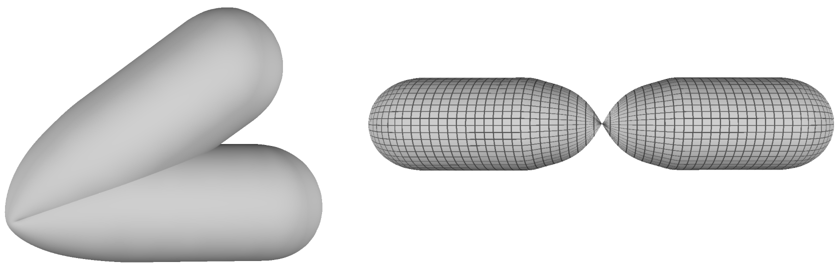
$$\begin{pmatrix} 1 & 0 & \vec{t_x} \\ 0 & 1 & \vec{t_y} \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(a) Translation (b) Homothétie (c) Rotation

FIGURE 8 – **Matrices de transformation du plan (2D)**

Notons $T_{\mathfrak{B}_i}$ les transformations relatives (depuis la pose d'origine) en chaque articulation. La transformation par le *Linear Bending Skinning* d'un sommet v du maillage d'origine est donnée comme l'interpolation des matrices de transformation par rapport aux poids associés en chaque sommet, c'est-à-dire que le sommet déformé est donné par la transformation suivante :

$$T(v) = \left(\sum_{i \in [\#\mathrm{os}]} w_{\mathfrak{B}_i}(v)\, T_{\mathfrak{B}_i} \right) \cdot v$$



Pli　　　　Torsion

Figure 9 – **Capsule déformée par Linear Bending Skinning**

Les principaux défauts dont souffre le *Linear Bending Skinning* sont liés à des problèmes de préservation de volume aux plis et aux torsions (figure 9). Ceci est dû au fait d'interpoler linéairement nos transformations, car l'interpolation d'une rotation correspondrait plus à une transformation vivant sur une sphère qu'une transformation en ligne droite tel que proposé.

4.3 Dual Quaternion Skinning

Une solution proposée est celle du *Dual Quaternion Skinning* [Kavan et al., 2007]. Cette méthode s'appuie sur l'utilisation des *quaternions* [Hamilton, 1848]. Les quaternions sont une généralisation des nombres complexes utilisée pour effectuer des rotations dans un espace 3D. À la différence qu'un nombre complexe s'écrit sous la forme algébrique $x + y\mathbf{i}$ alors qu'un quaternion s'écrit sous la forme $w + u_x\mathbf{i} + u_y\mathbf{j} + u_z\mathbf{k}$ avec $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. Notons qu'il est possible de noter la partie imaginaire du quaternion sous forme d'un vecteur $\vec{u} = \left(\begin{smallmatrix} u_x \\ u_y \\ u_z \end{smallmatrix}\right)$, par suite le quaternion sous la forme $w + \vec{u}$. La rotation d'une position $\vec{p} = \left(\begin{smallmatrix} x \\ y \\ z \end{smallmatrix}\right) = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ autour d'un axe dirigé par le vecteur unitaire $\vec{u}$ pour un angle de rotation θ et exprimée depuis l'utilisation d'un quaternion $q_\theta = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\vec{u}$ par le calcul suivant :

$$R(\vec{p}, \theta, \vec{u}) = q_\theta \cdot \vec{p} \cdot \bar{q}_\theta$$

où $\bar{q}_\theta = \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right)\vec{u}$ est le quaternion conjugué de q_θ.

La méthode *Dual Quaternion Skinning* propose ensuite depuis une idée similaire à celle du *Linear Bending Skinning* d'interpoler cette fois dans l'espace des quaternions :

$$q(v) = \frac{\sum_{i \in [\#\mathrm{os}]} w_{\mathfrak{B}_i}(v)\, q_{\mathfrak{B}_i}}{\left\| \sum_{i \in [\#\mathrm{os}]} w_{\mathfrak{B}_i}(v)\, q_{\mathfrak{B}_i} \right\|}$$

pour un sommet du maillage d'origine v et $q_{\mathfrak{B}_i}$ les quaternions associés aux transformations des os $\mathfrak{B}_i$. $q(v)$ et son conjugué $\bar{q}(v)$ sont ensuite utilisés pour effectuer la transformation du sommet v.

Bien que visuellement plus intéressante que le *Linear Bending Skinning* et presque aussi rapide, la méthode *Dual Quaternion Skinning* produit des effets de gonflements et d'auto-intersections du maillage lors de plis (figure 10). Ce que propose de corriger l'*Implicit skinning*.

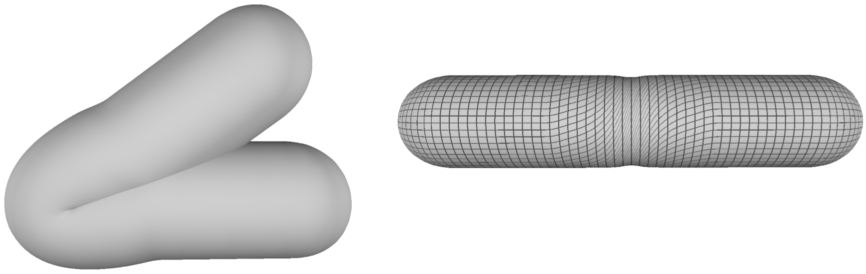


FIGURE 10 – **Capsule déformée par Dual Quaternion Skinning**

5 Surfaces implicites et composition

5.1 Surfaces implicites

Les surfaces implicites ont été initialement étudiées pour la fabrication assistée par ordinateur [Sabin, 1968]. Mais elles sont aussi plus récemment utilisées en modélisation [Zanni, 2013] et en animation [Vaillant et al., 2013]. À la différence des maillages qui offrent une représentation discrète d'une surface en 3D, une surface implicite est infiniment continue et souvent exprimée depuis une équation de la forme $f(P) = C$ où f induit un champ scalaire. Un champ scalaire est une fonction de $\mathbb{R}^d$ dans $\mathbb{R}$, avec d est la dimension de l'espace, qui associe à chaque position une valeur scalaire (bien souvent un réel). Pour s'imager un champ scalaire, on peut le voir par exemple comme un champ de températures qui donne une idée de la proximité à une région d'intérêt mais aussi d'orientation pour l'atteindre. La surface implicite est ensuite définie comme étant représentée par une des valeurs $C \in \mathbb{R}$ de ce champ $f : \mathbb{R}^d \mapsto \mathbb{R}$ et est implicitement donnée par les positions $P \in \mathbb{R}^d$ vérifiant l'équation $f(P) = C$. Ceci revient à dire qu'elle est donnée par l'image inverse de f pour C :

$$\mathcal{S}_{f,C} = f^{-1}(C) = \left\{P \in \mathbb{R}^d \mid f(P) = C\right\}.$$

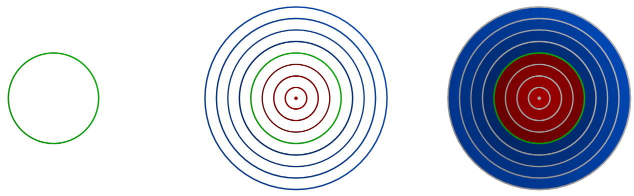
Par exemple, pour un cercle, f est la fonction de distance d'une position au centre du cercle et la valeur cible est le rayon du cercle : $\mathcal{C}(O, R) = \left\{P \in \mathbb{R}^2 \mid \|O - P\|_2 = R\right\}$.

L'un des avantages des surfaces implicites est que la normale en la surface est induite par le gradient du champ scalaire. Pour rappel le gradient $\forall P \in \mathbb{R}^3, \nabla f(P) = \left(\frac{\partial}{\partial x} f(P), \frac{\partial}{\partial y} f(P), \frac{\partial}{\partial z} f(P)\right)$ est calculé depuis les dérivées partielles de la fonction et donne la direction optimale vers laquelle aller pour maximiser notre fonction, c'est l'équivalent de la dérivée en plus grande dimension. La normale en un position $P \in \mathcal{S}_{f,C}$ de notre surface est donc donnée par :

$$\vec{\eta}_{f,C}(P) = -\frac{\nabla f(P)}{\|\nabla f(P)\|_2}.$$

Nous noterons une *iso-valeur* $I \in \mathbb{R}$ et son *iso-surface* associée $\mathcal{S}_{f,I} = f^{-1}(I)$. Si C est l'iso-valeur de notre iso-surface cible, nous noterons l'extérieur de la surface comme étant donné par l'ensemble iso-surfaces à iso-valeurs inférieures à C : $\left\{P \in \mathbb{R}^d \mid f(P) < C\right\}$ et l'intérieur par les iso-surfaces à iso-valeurs supérieures à C : $\left\{P \in \mathbb{R}^d \mid f(P) > C\right\}$. L'intérieur d'une surface implicite sera souvent représenté en rouge tel que présenté en figure11 (les plus grandes valeurs) et l'extérieur en bleu (les plus petites valeurs). Le support d'un champ scalaire est donné par l'ensemble des positions de l'espace à valeur de champ non-nulles : $\operatorname{supp}(f) = \left\{P \in \mathbb{R}^d \mid f(P) \neq 0\right\}$.

Bien qu'il existe plusieurs autres manières de définir et utiliser les surfaces implicites telles que par exemple les surfaces de convolution [Bloomenthal and Shoemake, 1991] [Jin and Tai, 2002], les *Hermite Radial Basis Functions* [Macedo et al., 2011], les surfaces définies depuis des squelettes [Angelidis and Cani, 2002], nous travaillerons par la suite sur des surfaces implicites à support borné représentant des champs de distance : ceci pour garder les valeurs non-nulles dans un espace borné et ignorer les positions



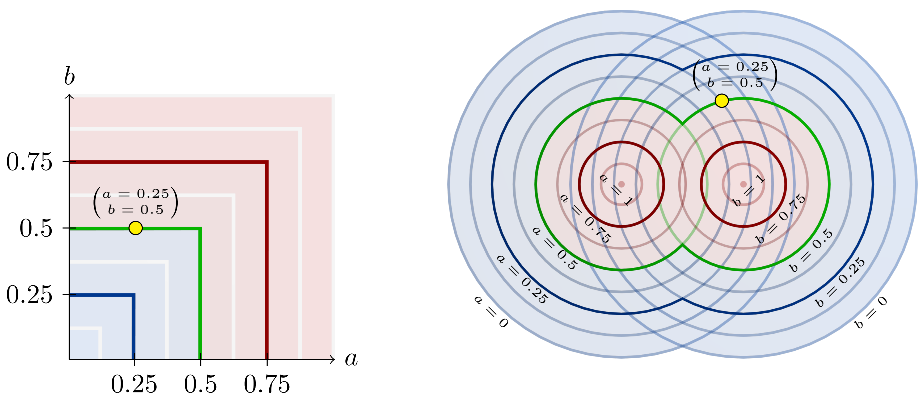
(a) surface (b) iso-surfaces (c) champ scalaire

FIGURE 11 – **Champ scalaire d'un cercle (surface implicite à support borné)**
(a) représente la surface continue issue du champ scalaire. (b) illustre des exemples d'iso-surfaces (iso-valeurs : 1 au centre, 0.875, 0.75, 0.625, 0.5 surface cible, 0.375, 0.25, 0.125, frontière avec 0). (c) est le support du champ scalaire : l'ensemble infini des iso-surfaces d'iso-valeur non nulle. En bleu on représente les iso-surfaces à l'extérieur de la surface (en vert) et en rouge celles à l'intérieur de la surface.

à une distance trop importante de notre surface. Par convention les valeurs du champ vivront dans $[0, 1]$ et valent 0 passé une certaine distance comme illustré en figure 11. De même l'iso-valeur cible sera 0.5 et la valeur en le squelette topologique de la surface sera 1. L'avantage de cette convention est de pouvoir composer nos surfaces à l'aide d'opérateurs à support borné.

5.2 Opérateurs de composition

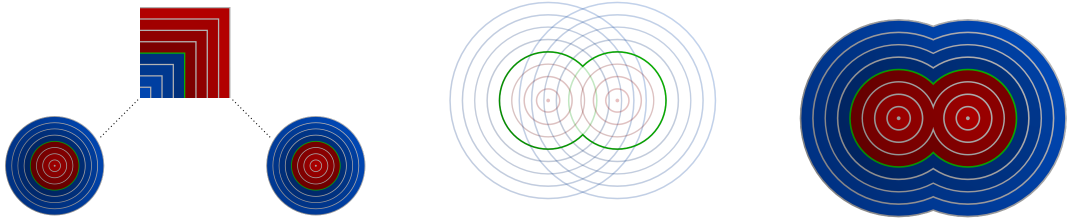
L'une des propriété intéressante des surfaces implicites est la possibilité de les composer entre elles. C'est-à-dire qu'en utilisant un opérateur entre plusieurs surfaces implicites, il est possible de les composer pour obtenir une nouvelle surface implicite. Cette idée s'inscrit dans le concept de *Constructive Solid Geometry* [Requicha, 1980] qui donne la construction d'une forme depuis un arbre de composition à l'aide d'opérateurs boolean. Dans le cas des surfaces implicites, ceci fonctionne sur le même principe par le concept *BlobTree* [Wyvill et al., 1986] avec des opérateurs dédiés à la composition du champ scalaire et des déformations de l'espace (torsions, rotations, homothéties, translations).



(a) opérateur (b) iso-surfaces (0.25, 0.5 et 0.75)

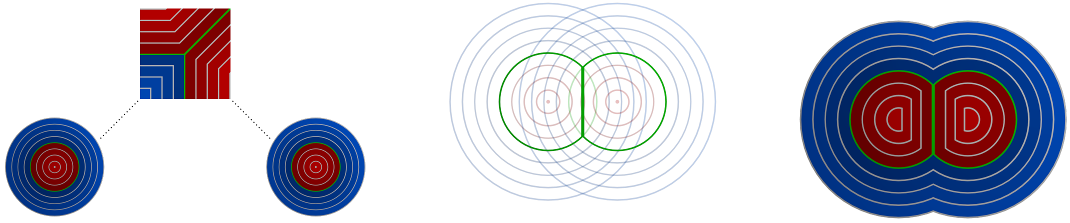
FIGURE 12 – **Lecture graphique de la composition par l'opérateur max**
Il est possible de lire graphiquement (a) la valeur de l'opérateur depuis les valeurs des deux surfaces à composer : par exemple lorsque la valeur est 0.25 *en a et* 0.5 *en b, l'opérateur max retourne comme valeur scalaire* $\max(0.25, 0.5) = 0.5$. *De plus il est possible de lire visuellement les iso-surfaces depuis l'opérateur : on représente sur l'opérateur et dans la composition résultante (b) les iso-surfaces des valeurs* 0.25, 0.5 *et* 0.75.

Un opérateur binaire *op* sur deux champs scalaires $a : \mathbb{R}^d \mapsto \mathbb{R}$ et $b : \mathbb{R}^d \mapsto \mathbb{R}$ donne un nouveau



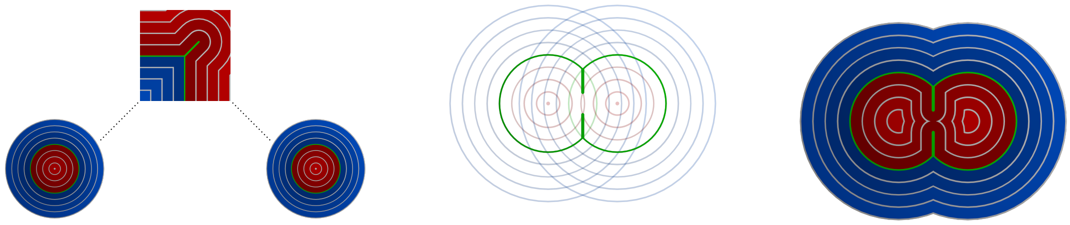
composition iso-surface champ scalaire

(a) Composition par l'opérateur max



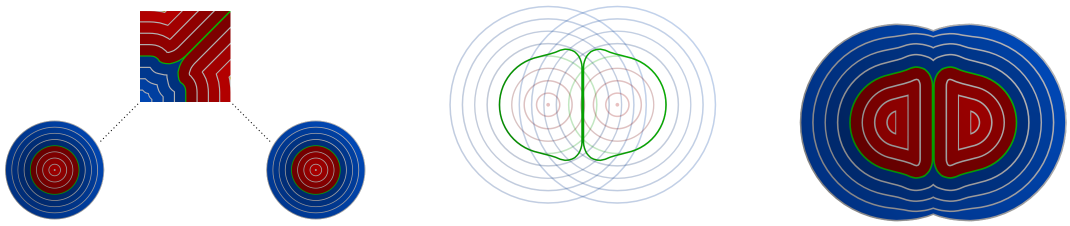
composition iso-surface champ scalaire

(b) Composition par l'opérateur de contact en champ de distance



composition iso-surface champ scalaire

(c) Composition par l'opérateur de contact partiel en champ de distance



composition iso-surface champ scalaire

(d) Composition par un opérateur de gonflement

FIGURE 13 – **Examples de compositions de sphères par des opérateurs binaires**

En premier, le nœud dans un BlobTree (les surfaces et la représentation graphique de l'opérateur), puis la surface obtenue depuis la composition et le champ scalaire final de la nouvelle surface.

champ scalaire issu de la composition des champs a et b :

$$op\left(a,b\right):\mathbb{R}^d\mapsto\mathbb{R}.$$

La surface implicite issue de la composition est ensuite exprimée de la même manière que précédemment :

$$\mathcal{S}_{op(a,b),C}=\left(op\left(a,b\right)\right)^{-1}\left(C\right)=\left\{P\in\mathbb{R}^d\mid op\left(a,b\right)\left(P\right)=C\right\}.$$

Le gradient dans ce nouveau champ scalaire est ensuite exprimé depuis les gradients en a et en b (théorème de dérivation des fonctions composées) par :

$$\nabla op\left(a,b\right)=\left(\frac{\partial}{\partial a}op\left(a,b\right)\right)\nabla a+\left(\frac{\partial}{\partial b}op\left(a,b\right)\right)\nabla b.$$

Un exemple d'opérateur simple est l'opérateur $\max\left(a,b\right):P\mapsto\max\left(a\left(P\right),b\left(P\right)\right)$ qui permet l'union de deux surfaces implicites (figure 12 et figure 13 (a)). L'opérateur d'union propre [Pasko et al., 1995] permet de rester $\mathcal{C}^1$ (il n'existe ni discontinuité de valeur dans le champ et ni dans le gradient). Des travaux s'intéressent à conserver un champ propre [Canezin et al., 2013] pour la composition. Mais l'intérêt de l'opérateur max pour nos travaux est qu'il conserve les champs de distance.

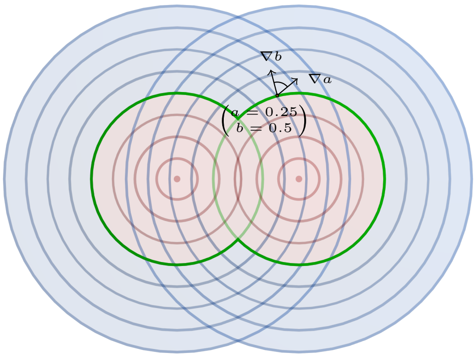


FIGURE 14 – **Angle entre les gradients de deux champs scalaires**

En informatique, nous préférons travailler sur des opérateurs bornés, car pour des opérateurs plus coûteux en temps de calcul, il est possible de pré-calculer des valeurs significatives pour ensuite interpoler les autres valeurs lors de l'évaluation de l'opérateur. De plus, les opérateurs binaires à support borné peuvent être représentés comme illustré en figure 12.

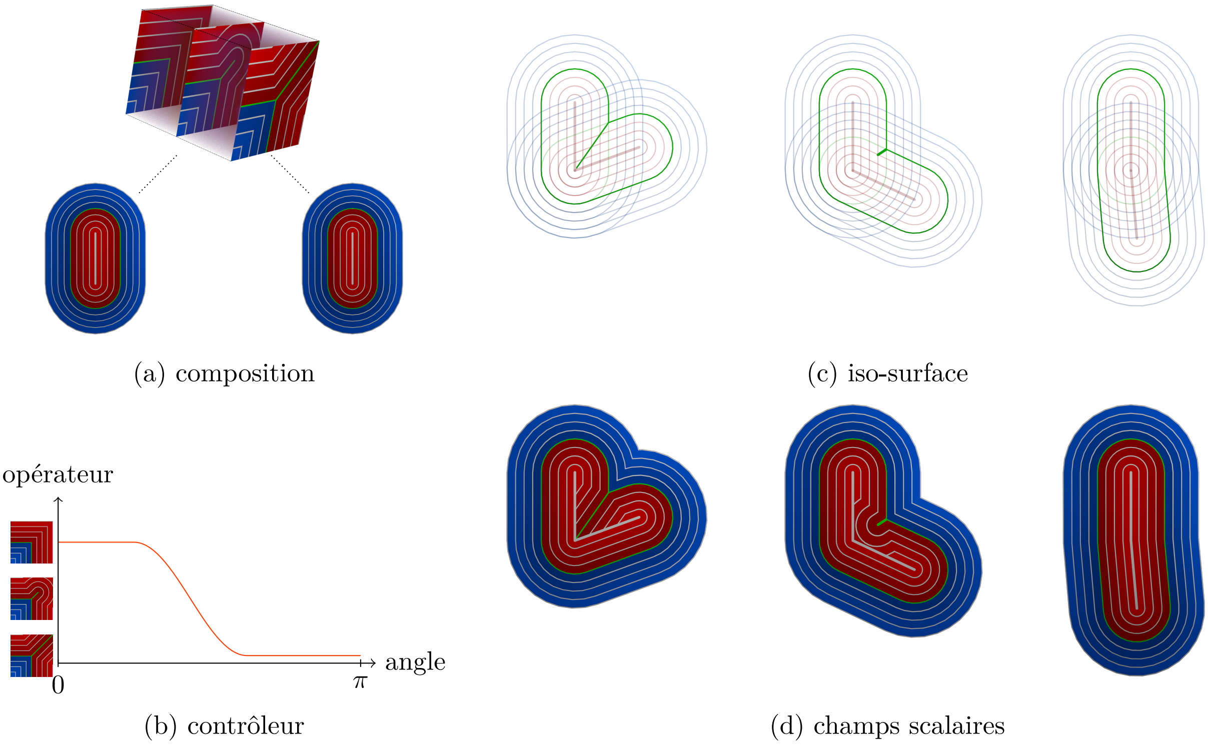
Par la suite nous nous intéresserons aux opérateurs basés champs de distance. La figure 13 donne des exemples d'opérateurs en champ de distance comme l'opérateur max (a), l'opérateur de contact (b) et l'opérateur de contact partiel (c). De plus, je donne un aperçu de ce que pourrait être un opérateur de grossissement en champ de distance (d). Ces opérateurs [Vaillant et al., 2013] sont à l'origine fabriqués depuis le résultat souhaité pour la surface implicite et par interpolation biharmonique pour les autres iso-valeurs.

5.3 Opérateurs binaires basés gradient

Les opérateurs binaires basés gradient permettent une composition entre deux champs scalaires en prenant en plus l'information de l'angle non signé vivant dans $[0,\pi]$ entre les gradients de ces deux champs (figure 14) :

$$(\widehat{\nabla a,\nabla b})=\cos^{-1}\left(\frac{\nabla a\cdot\nabla b}{\left\|\nabla a\right\|_2\left\|\nabla b\right\|_2}\right).$$

L'intérêt de ces opérateurs est qu'ils offrent un meilleur paramétrage pour la modélisation [Gourmel et al., 2013] par la surfaces implicites et ont grandement contribué à la naissance de l'Implicit skinning [Vaillant et al., 2013]. [Angles et al., 2017] propose de les construire graphiquement depuis une interface utilisateur.



(a) composition

(b) contrôleur

(c) iso-surface

(d) champs scalaires

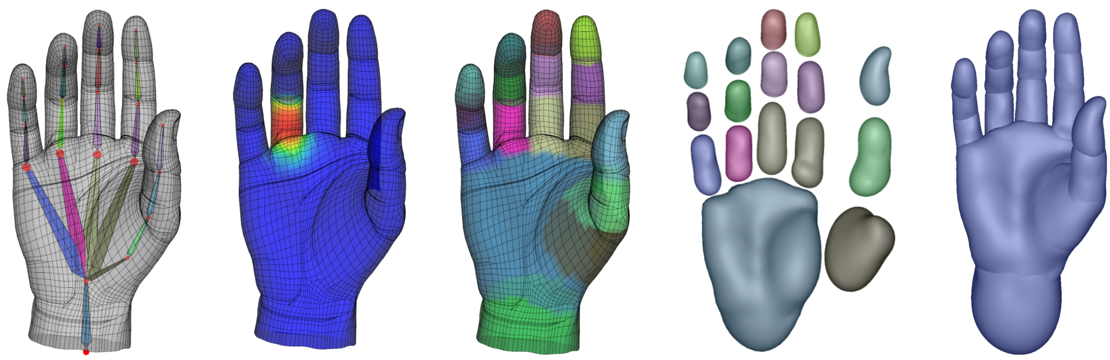
FIGURE 15 – **Composition de capsules par un opérateur de contact basé gradient**
En (a) une représentation de la composition de deux capsules dans le BlobTree par un opérateur de contact en champs de distance. En (b) le contrôleur de l'opérateur basé gradient : selon la valeur de l'angle entre les gradients en une position, on utilise une certaine tranche de l'opérateur basé gradient ce qui donne un opérateur binaire pour composer les valeurs scalaire en cette position. En (c) est représenté la surface implicite obtenue par composition par l'opérateur de contact et en (d) le champ scalaire résultant pour la composition de capsules dans différentes positions.

L'opérateur basé gradient est un opérateur en 3 dimensions dont deux sont les valeurs scalaires et la troisième un paramètre. Ce paramètre va permettre de choisir l'opérateur à utiliser. L'opérateur peut être pré-calculé dans une grille de résolution $128 \times 128 \times 128$ puis reconstruit lors de l'évaluation depuis par interpolation trilinéaire des valeurs échantillonnées. Le paramètre est ensuite obtenu depuis une fonction de contrôle qui prend en entrée l'angle entre les gradient tel qu'illustré en figure 15 (b).

6 Correction par l'Implicit skinning

6.1 Segmentation en surfaces implicites

L'Implicit skinning passe par une première étape de pré-calcul des surfaces implicites (figure 16). Ces surfaces implicites sont calculées depuis l'information donnée par les poids d'animation définis lors de l'étape du *rigging*. Depuis ces poids, on forme une partition des sommets du maillage où est associé à chaque os le sous-maillage des sommets dont il est l'os avec le poids d'influence le plus important. Depuis ce sous-maillage, on obtient des positions S_i et leur normale $\vec{\eta}_i$ depuis un échantillonnage aléatoire [Bridson, 2007] de la surface discrète. L'échantillonage aléatoire donne en général de bons résultats et il serait trop contraignant d'imposer à la surface d'interpoler les sommets. La surface implicite est ensuite calculée depuis cet échantillonnage par les *Hermite Radial Basis Functions* [Macedo et al., 2011]. Ceci demande l'utilisation d'une fonction de base $\phi : \mathbb{R} \mapsto \mathbb{R}$, dans le cas de l'Implicit skinning $\phi : x \mapsto x^3$.



(a) (b) (c) (d) (e)

(Images de [Vaillant et al., 2013])

FIGURE 16 – **Segmentation en surfaces implicites pour l'Implicit skinning**
Le maillage d'une main et son squelette d'animation (a). Les poids de skinning associé à une articulation (b) tel qu'illustré en figure 7. Les sommets du maillage sont segmentés en groupes (c) selon les informations données par les poids de skinning. Chaque groupe est associé à un os du squelette d'animation. Pour chaque groupe, on calcule une surface implicite (d) : Hermite Radial Basis Functions [Macedo et al., 2011]. (e) illustre la composition des surfaces implicites par l'opérateur d'union propre [Pasko et al., 1995].

Ensuite le champ scalaire est donné par :

$$d_{\mathfrak{B}_i}(P) = \sum_{i \in [\#\text{échantillons}]} \Big(\alpha_i \phi \left(\|P - S_i\| \right) + \vec{\beta}_i \cdot \nabla \phi \left(\|P - S_i\| \right) \Big) .$$

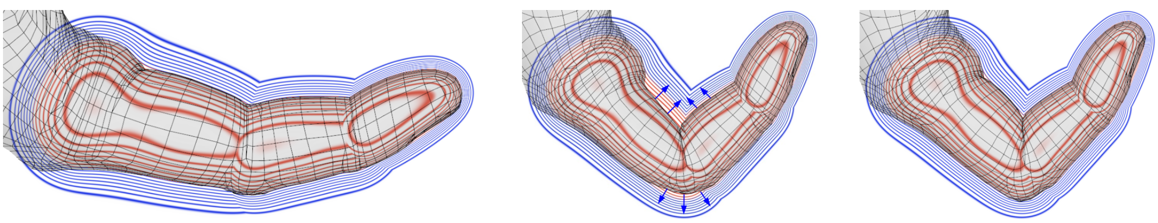
Où les scalaires α_i et les vecteurs $\vec{\beta}_i$ sont calculés depuis la résolution du système linéaire dont les contraintes sont données sur les échantillons S_i et leur normale $\vec{\eta}_i$ par :

$$\begin{cases} d_{\mathfrak{B}_i}(S_i) = 0 \\ \nabla d_{\mathfrak{B}_i}(S_i) = \vec{\eta}_i \end{cases} .$$

Dans la version actuelle de l'Implicit skinning, les *Hermite Radial Basis Functions* ont été remplacées par des champs de distance signés exprimés comme la distance à l'iso-surface des *Hermite Radial Basis Functions*. Pour chaque sommet, on associera la valeur du champ scalaire en sa position d'origine pour l'y projeter lors de l'étape du tracking.

6.2 Correction du maillage

Avant l'utilisation du procédé de l'Implicit skinning, **la déformation du maillage** utilise la méthode ***Dual Quaternion Skinning*** étudiée précédemment pour obtenir la déformation globale des sommets avant correction.



Maillage d'origine Maillage déformé Maillage projeté

(Images de [Vaillant et al., 2013])

FIGURE 17 – **Étape de projection des sommets sur leur iso-surface d'origine**

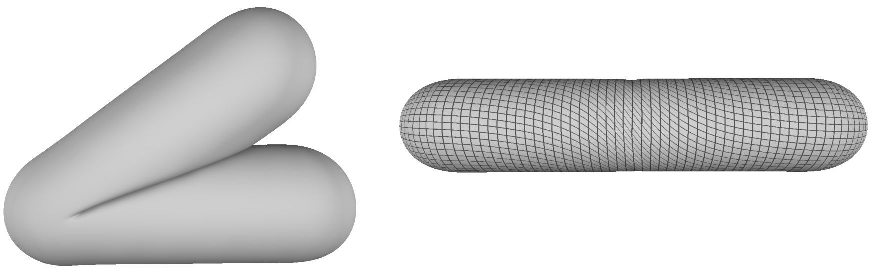
Ensuite le champ scalaire utilisé est fabriqué depuis la composition des surfaces implicites. Le *BlobTree* utilisé pour la composition est donné par le squelette d'animation : il induit les transformation de l'espace en chaque os et l'ordre de composition depuis sa structure d'arbre. Précédemment l'opérateur utilisé était un opérateur de contact fabriqué depuis un interpolation biharmonique. Mais dans la version sur laquelle je travaille, l'opérateur de composition utilisé est un opérateur de contact en champs de distance tel qu'illustré en figure 15.

La correction par l'Implicit skinning suit ensuite les étapes suivantes :

— Dans un premier temps vient l'étape de **projection des sommets déformés sur leur iso-surface d'origine** (figure 17). Ceci permet par exemple de conserver des détails du maillage.

— Ensuite vient l'étape de **relaxation tangentielle** dont le but est de conserver la consistance du maillage : éviter une déformation excessive des faces due à des déformations trop importantes des sommets.

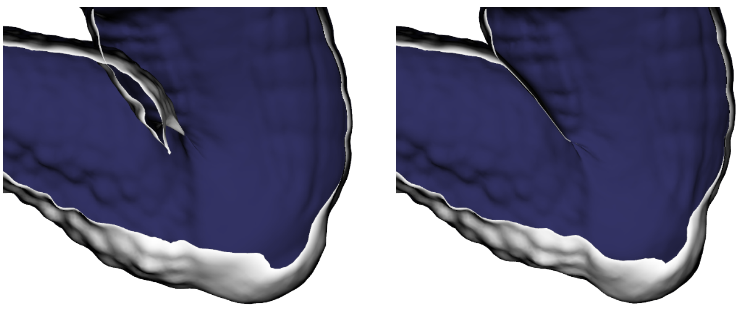
— Puis vient l'étape de **lissage de la surface** par un lissage **Laplacien** pour adoucir les plis et zones de contact et éviter un maillage trop épineux (on suppose que le maillage d'origine est relativement lisse et donc il n'y a pas de raison d'avoir des pointes trop importantes).



Pli Torsion

FIGURE 18 – **Capsule déformée par l'Implicit Skinning**

L'Implicit skinning permet en particulier de proposer une correction intéressante aux plis d'un maillage et reste robuste lors de la torsion d'une articulation (figure 18). De plus l'Implicit skinning est aussi intéressant car il permet de réaliser des auto-intersections propres aussi bien aux plis qu'aux contacts du maillage avec lui-même (figure 19).



Dual Quaternion Skinning [Kavan et al., 2007] *Implicit skinning* [Vaillant et al., 2013]

(Images de [Vaillant et al., 2013])

FIGURE 19 – **Vue en coupe de la correction d'un pli lors d'une auto-intersection**

6.3 Améliorations de la méthode

On pose I_i l'iso-value dans le champ avant déformation du sommet v_i du maillage d'origine. [Vaillant et al., 2013] propose de déformer v_i en $\widehat{v_i}$ puis de résoudre numériquement $f\left(\widehat{v_i}\right) - I_i$ et itérer jusqu'à convergence. Aujourd'hui, l'idée est de profiter du fait d'utiliser un champ de distance (les opérateurs compositions préservent l'aspect champ de distance) pour effectuer cette étape de projection en une seule itération :

$$\widehat{v_i} \leftarrow \widehat{v_i} + \left(f\left(\widehat{v_i}\right) - I_i\right) \frac{\nabla f\left(\widehat{v_i}\right)}{\left\|\nabla f\left(\widehat{v_i}\right)\right\|_2}.$$

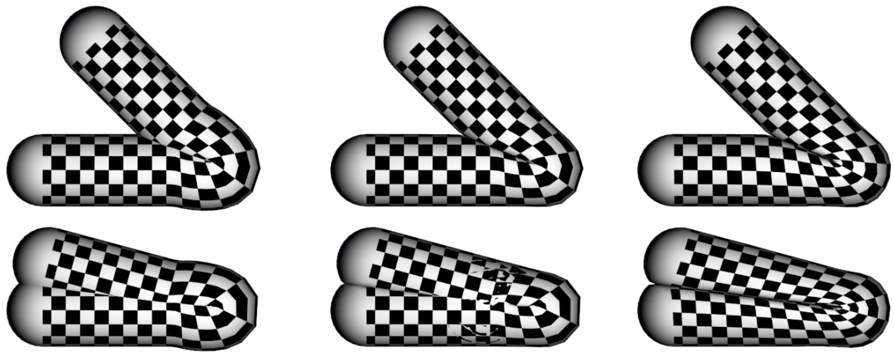
Les principales améliorations de [Vaillant et al., 2013] à [Vaillant et al., 2014] sont le changement de la relaxation tangentielle et le passage à une méthode incrémentale.

Pour la relaxation tangentielle [Vaillant et al., 2013] propose de considérer chaque sommet dans le polygone formé par son voisinage (les sommets adjacents : liés par une arête), de projeter ce polygone dans le plan tangent à ce sommet et y considérer ses coordonnées barycentriques généralisées [Hormann and Floater, 2006]. Ensuite la relaxation tangentielle consiste à itérer de sorte de déplacer chaque sommet dans son plan tangent local avec comme compromis que se rapprocher de la position estimée depuis les coordonnées barycentriques doit rapprocher de l'iso-valeur d'origine.

[Vaillant et al., 2014] propose de remplacer cette méthode par la minimisation de l'énergie ARAP (As-Rigid-As-Possible) [Sorkine-Hornung and Alexa, 2007]. Cette énergie se base sur les rotations de l'espace pour conserver autant que possible la forme du maillage sur les zones rigides tout restant localement similaire et flexible sur les parties déformées. Pour cela, pour chaque sommet v_i on se place dans son voisinage $\mathcal{N}(v_i) = \{v_k\}_k$. Nous considérerons les angles $\alpha_{i,k} = (\widehat{\overrightarrow{v_{k-1}v_i}, \overrightarrow{v_{k-1}v_k}})$ et $\beta_{i,k} = (\widehat{\overrightarrow{v_{k+1}v_i}, \overrightarrow{v_{k+1}v_k}})$ pour déterminer les poids cotangent impliqués par la suite. Pour la calcul de l'énergie ARAP, on suppose que l'on connaît la matrice de rotation R_i en chaque sommet. Dans le cas de l'Implicit skinning, elles sont données par la déformation du squelette d'animation. Le but étant ensuite de comparer la différence locale entre les sommets déformés par le skinning et la transformation locale du maillage d'origine par la matrice rotation. Autrement dit, on s'intéresse à minimiser la différence locale en v_i entre les arêtes déformées $(\widehat{v_i} - \widehat{v_k})$ et la déformation des arêtes $R_i(v_i - v_k)$ de $\mathtt{star}(v_i)$ par R_i. D'où la minimisation de l'énergie ARAP :

$$\sum_{i \in [\#sommets]} \sum_{v_k \in \mathcal{N}(v_i)} \frac{\cot(\alpha_{i,k}) + \cot(\beta_{i,k})}{2} \left\|(\widehat{v_i} - \widehat{v_k}) + R_i(v_i - v_k)\right\|^2.$$

La méthode actuelle de l'Implicit skinning est incrémentale, c'est-à-dire qu'à la différence de [Vaillant et al., 2013] qui transforme la maillage d'origine par le *Dual Quaternion Skinning* puis le corrige, [Vaillant et al., 2014] propose de transformer le maillage depuis la pose précédente en considérant les transformations relatives à celle-ci puis de le corriger. Ceci permet comme illustré en figure 20 de corriger des erreurs de projection. Car dans le cas de déformations trop larges, un sommet peut être projeté de l'autre côté de la surface ce qui n'est pas souhaité. De plus, l'énergie ARAP permet de mieux répartir les sommets sur la surface en particulier sur les parties coudées du maillage lors de déformations importantes.



Dual Quaternion Skinning [Kavan et al., 2007] — *Implicit skinning 2013* [Vaillant et al., 2013] — *Implicit skinning 2014* [Vaillant et al., 2014]

(Images de [Vaillant et al., 2014])

FIGURE 20 – **Amélioration de l'Implicit skinning**

Troisième partie

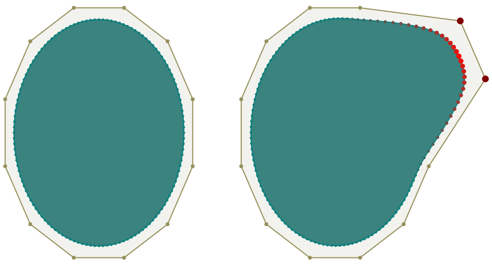
Déformations de surfaces implicites par les cages

J'expliquerai le fonctionnement des déformations basées cages et ce qui en fait un problème de recherche dans le cas des surfaces implicites. Puis, je présenterai les méthodes que nous avons essayé pour résoudre ce problème et commenterai leur pertinence.

7 Cages de déformation et problème de la position inverse

7.1 Cages de déformation et coordonnées barycentriques généralisées

Une cage de déformation est une surface discrète en 2D ou en 3D (maillage). C'est en général une simplification de notre surface d'intérêt qui définit des positions de contrôle aux régions que l'on souhaite déformer (figure 21). Une cage englobe la surface d'intérêt et produit des déformations plus libres et adoucies de notre surface que pourrait le faire un squelette d'animation. Bien qu'elle puisse être définie de manière semi-automatique [Casti et al., 2019], nous admettrons qu'elle est conçue par un infographiste et donc par conséquent quelconque bien qu'elle doit rester cohérente pour l'animation (par exemple, elle ne s'auto-intersecte pas en pose d'origine).



Disque et sa cage Déformation du disque

FIGURE 21 – **Exemple d'un disque et sa déformation par une cage**

Une cage fait l'interpolation de l'espace qu'elle englobe et propose l'interpolation d'une portion de l'espace déformé qu'elle englobe depuis la correspondance avec son état d'origine. L'exemple plus simple de cage est le triangle, en particulier que ses coordonnées barycentriques sont les coordonnées barycentriques [Möbius, 1827] telles que nous les connaissons qui proposent une interpolation linéaire dans le triangle (En annexe A.1 que j'invite le lecteur à regarder pour comprendre ensuite leur généralisation aux cages).

La déformation de l'espace par la cage sera ensuite obtenue depuis l'utilisation de coordonnées barycentriques généralisées. L'avantage des cages de déformation est de pouvoir choisir un système de coordonnées barycentriques en fonction de la déformation souhaitée ce qu'on nomme le *binding* dont le procédé consiste à transformer le système de coordonnées cartésiennes en un système de coordonnées barycentriques propre à notre cage. Le *binding* est l'équivalent aux cages qu'est le *rigging* aux squelettes d'animation. Cette étape est obligatoire pour utiliser une cage de déformation car le *binding* lie chaque position de l'espace d'origine à sa position dans l'espace déformé. Cette étape peut être longue à calculer selon le système de coordonnées choisi et fabrique les poids associés aux sommets de notre cage.

Pour une cage $\{C_i\}_{i\in[n]}$, ces poids $(\alpha_i(P))_{i\in[n]}$ sont calculés pour une position P dans l'espace de coordonnées cartésiennes tels que :

$$P = \sum_{i\in[n]} \alpha_i(P) C_i.$$

De plus, on impose $\sum_{i\in[n]} \alpha_i(P) = 1$, ce qui en fait des coordonnées barycentriques. L'avantage des coordonnées barycentriques est qu'elles sont invariantes par transformation affine (prouvé en annexe

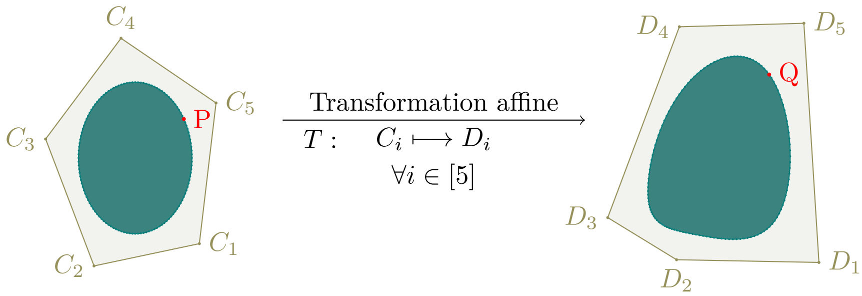


FIGURE 22 – **Déformation d'un disque par une cage pentagonale**
P est bindé dans la cage $\{C_i\}_{i\in[5]}$ par le calcul des poids $(\alpha_i(P))_{i\in[5]}$. $P = \sum_{i\in[5]} \alpha_i(P)C_i$ est ensuite transformé par la transformation affine T telle que $T(P) = \sum_{i\in[5]} \alpha_i(P)D_i = Q$.

A.2), autrement dit une transformation affine est linéaire par rapport aux coordonnées barycentriques. Soit $T : C_i \longmapsto D_i$ la transformation affine qui envoie les sommets $\{C_i\}_{i\in[n]}$ de la cage dans l'espace d'origine vers les sommets $\{D_i\}_{i\in[n]}$ de la cage correspondante dans l'espace déformé. Ceci revient à dire qu'on a translaté les C_i en leur nouvelle position D_i et c'est de cette manière que la cage sera déformée par l'infographiste. Puisque T est une transformation affine, $Q = T(P)$ peut être évalué par une simple combinaison linéaire des D_i (illustration en figure 22) :

$$Q = T(P) = T\left(\sum_{i\in[n]} \alpha_i(P)C_i\right),$$

par invariance affine des coordonnées barycentriques (annexe A.2) :

$$Q = \sum_{i\in[n]} \alpha_i(P)T(C_i),$$

puis par définition de T :

$$Q = \sum_{i\in[n]} \alpha_i(P)D_i.$$

Il existe de nombreux système de coordonnées barycentriques généralisées aux cages de déformation présentant chacun leurs avantages et leurs inconvénients. Historiquement, le premier système de coordonnées utilisé pour les cages sont les *Wachspress Coordinates* [Wachspress, 1975] [Meyer et al., 2002] qui offrent une interpolation linéaire de l'espace mais souffre de problèmes de continuité lors de la création de concavités. Les *Mean Value Coordinates* [Floater, 2003] [Ju et al., 2005] [Hormann and Floater, 2006] sont définies pour des types de cages et déformations quelconques et de plus permettent l'interpolation aussi à l'extérieur de la cage. Cependant, pour des cages à design concave, seuls les sommets dans le noyau de la cage sont à poids positifs, dans le reste de l'espace, les poids en certaines coordonnées peuvent être négatifs, par conséquent, ils ont une influence. Cette déformation non esthétique est corrigée par les *Harmonic Coordinates* [Joshi et al., 2007] proposées par Pixar, dont le principe est d'imposer pour conditions de vérifier l'équation de Laplace à l'intérieur de la cage et aussi d'être $\mathcal{C}^0$ aux bordures de la cage (respecter un système de coordonnées barycentrique 2D sur les faces de la cage : nous travaillons en général sur des faces triangulaires) en résolvant le problème de Dirichlet.

D'autres système de coordonnées ont aussi vu le jour comme les *Green Coordinates* [Lipman et al., 2008] dont l'intérêt est de proposer une déformation par cage conforme et 2D et quasi-conforme en 3D. Une déformation conforme est une déformation qui préserve localement les angles ce qui implique des déformations de l'espace par les cages plus rigides. En effet, l'idée de la transformation conforme est d'envoyer des sphères infinitésimales d'un espace vers des sphères infinitésimales dans un autre espace. Sur cette même idée, une déformation quasi-conforme envoie des sphères infinitésimales vers des ellipsoïdes infinitésimales ce qui produit malgré tout des déformations de l'espace assez rigides.

Les systèmes de coordonnées barycentriques sont nombreux, on trouve aussi bien des améliorations des *Mean Value Coordinates* définis pour des faces non nécessairement triangulaires [Langer et al., 2006], sans poids négatifs [Lipman et al., 2007], avec une interpolation cubique [Li et al., 2013]. Ceci est aussi valable pour les fonctions harmoniques généralisées avec les *Biharmonic coordinates* [Weber et al., 2012] ce qui offre une plus grande élasticité à l'intérieur de la cage par la résolution de l'équation biharmonique. Ainsi que des coordonnées barycentriques plus locales [Zhang et al., 2014] depuis la résolution d'un problème d'optimisation (à noter que les déformations par les cages restent des déformations globales). De même, on il existe aussi des systèmes avec de bonnes propriétés et un *binding* assez rapide pour proposer la modification de positions de contrôle en temps-réel [Wang et al., 2015]. On trouve aussi plus récemment un système de coordonnées basé physique [Savoye, 2017] : sur la dynamique des fluides.

Il existe de même des travaux sur les **Cages* [García et al., 2013] qui permettent l'utilisation de cages multiples et de systèmes de coordonnées barycentriques différents.

7.2 Problème de la position inverse pour les surfaces implicites

Nous aimerions utiliser les cages de déformation pour déformer des surfaces implicites. Ceci entre autre pour pouvoir utiliser ce type de déformation avec l'Implicit skinning. Au moment de mon stage, il n'existe pas de méthode permettant la déformation de surface implicite par une cage de déformation telle que présentée précédemment. Cependant, des travaux existent sur la déformation d'une surface implicite par les *Free-Form Deformations* [Sederberg and Parry, 1986] qui sont un cas particulier de cages dont le maillage d'origine est un parallélépipède (qu'il est possible de subdiviser pour obtenir des positions de contrôle supplémentaire).

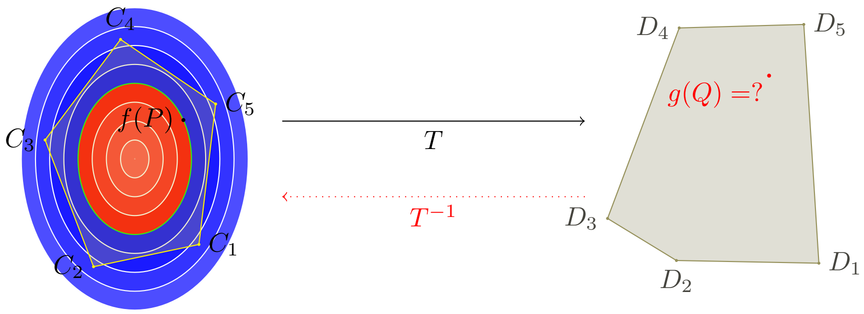


FIGURE 23 – **Problème de la position inverse pour l'évaluation du champ scalaire**
f définit un champ scalaire dont la surface implicite sera déformée par la cage $\{C_i\}_{i\in[5]}$. $\{C_i\}_{i\in[5]}$ est déformée en $\{D_i\}_{i\in[5]}$ par T, on évalue le champ scalaire g en une position Q dans l'espace déformé par T. L'évaluation de $g(Q)$ revient à connaître la position P dans l'espace d'origine tel que $T(P)=Q$, pour obtenir $g(Q)=f(P)$.

Le problème de la déformation d'une surface implicite est posé comme étant le fait que nous avons une cage de déformation en pose d'origine $\{C_i\}_{i\in[n]}$ et une surface implicite définie par un champ scalaire f. La déformation de la cage $\{C_i\}_{i\in[n]}$ vers une pose définie par $\{D_i\}_{i\in[n]}$ déforme le champ scalaire f en un champ scalaire g. Ceci implique que pour une position P dans l'espace d'origine, on a $g(T(P))=f(P)$. Or, ce que l'on cherche c'est à évaluer le champ scalaire g en toute position de l'espace déformé. Pour ceci, lorsqu'on évalue g en une position Q dans l'espace déformé, ceci demande de trouver P dans l'espace d'origine tel que son image par la transformation affine T soit la position Q. Ce qui nous donnera la valeur du champ scalaire en g : $g(Q)=g(T(P))=f(P)$.

Malheureusement, dans le cas de cages autres que les triangles et les tétrahèdres, la transformation affine n'est pas bijective. Elle est seulement surjective car l'espace affine $D_1 \times \mathtt{vect}\left\{\overrightarrow{D_1D_i}\right\}_{i\in[\![2,n]\!]}$ est de partie vectorielle formée par une famille liée (il existe des combinaisons linéaires de vecteurs entre eux). Par conséquent une position dans l'espace déformé n'a pas une unique écriture, par suite n'a pas un unique antécédent dans $C_1 \times \mathtt{vect}\left\{\overrightarrow{C_1C_i}\right\}_{i\in[\![2,n]\!]}$. Le problème posé ici est celui de retrouver la position inverse (illustré en figure 23) d'une position et fait de la déformation des surfaces implicites par les cages un problème de recherche.

7.3 *Free-Form Deformations* et résolution du problème de la position inverse

Les *Free-Form Deformations* [Sederberg and Parry, 1986] sont un type de cage particulier basé sur l'utilisation d'un parallélépipède comme cage d'origine. L'intérêt de cette cage est d'y définir des coordonnées à 2 variables en 2D et à 3 variables en 3D qui sont simplement les coordonnées définies par une base vecteurs du parallélépipède dans l'espace d'origine. Puis depuis ces coordonnées, on définit la transformation induite par la cage comme étant donnée par un volume de Bézier (produit tensoriel multivarié des polynômes de Bernstein). Ceci offre la possibilité de définir des positions de contrôle sur la parallélépipède par sa subdivision et de produire une déformation $\mathcal{C}^\infty$ de l'espace par une transformation affine des positions (à noter que les polynômes de Bernstein forment un base de coordonnées barycentriques). Pour les $\{P_{i,j,k}\}_{(i,j,k)\in[l]\times[m]\times[n]}$ positions de contrôle et (s,t,u) les coordonnées d'une position P dans le repère donné par le parallélépipède, dont la transformation est donnée par :

$$\mathcal{B}(s,t,u) = \sum_{(i,j,k)\in[l]\times[m]\times[n]} \binom{l}{i}\binom{m}{j}\binom{n}{l} s^i (1-s)^{l-i} t^j (1-t)^{m-j} u^k (1-u)^{n-k} T(P_{i,j,k})$$

où T est la transformation affine des $P_{i,j,k}$ et $\mathcal{B}$ la reconstruction de la position par le volume de Bézier. À noter qu'une volume de Bézier tel quel est un approximant, ce qui signifie en général $\mathcal{B}(\mu(P_{i,j,k})) \neq T(P_{i,j,k})$ où μ donne les coordonnées (s,t,u) d'une position dans la cage d'origine.

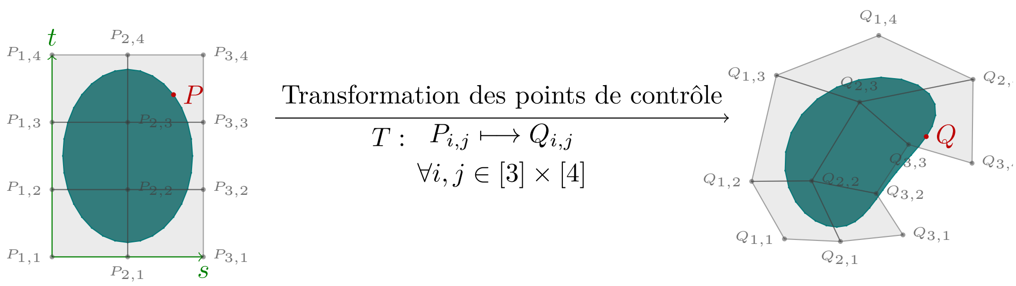


FIGURE 24 – **Déformation d'un disque par *Free-Form Deformations***

Tout comme les déformations basées cage quelconque, cette déformation n'est pas inversible et demande de résoudre le problème de la position inverse. [Parry, 1986] propose de résoudre ce problème par une résolution numérique. En effet, pour une position Q de l'espace déformé, ceci consiste à sélectionner la position $T(P_{i,j,k})$ la plus proche de Q puis de résoudre numériquement le problème en utilisant par exemple la méthode de Newton.

La méthode de Newton, est une méthode itérative de résolution numérique qui permet de trouver les zéros d'une fonction. Son principe est de suivre la direction optimale pour atteindre un zéro dans la même idée que ce que l'on peut faire en suivant le gradient d'une fonction. Dans notre cas nous souhaitons trouver (s,t,u) tels que $(\mathcal{B}(s,t,u) - Q) = 0$. La méthode de Newton propose d'utiliser la matrice Jacobienne de la transformation $\mathcal{B}$, c'est la matrice donnée par les dérivées partielles en chaque variable :

$$J_{\mathcal{B}}(s,t,u) = \left(\frac{\partial}{\partial s}\mathcal{B}(s,t,u), \frac{\partial}{\partial t}\mathcal{B}(s,t,u), \frac{\partial}{\partial u}\mathcal{B}(s,t,u)\right).$$

Les coordonnées de départ sont données en utilisant le $T(P_{i,j,k})$ le plus proche par $\lambda_0 = (s_0, t_0, u_0) = \left(\frac{i}{l}, \frac{j}{m}, \frac{k}{n}\right)$. Puis chaque itération est donnée par :

$$\lambda_{p+1} = \lambda_p + J_{\mathcal{B}}(\lambda_p)^{-1}(\mathcal{B}(\lambda_p) - Q).$$

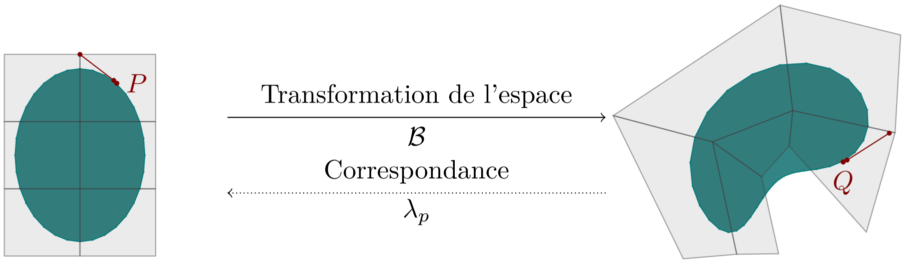
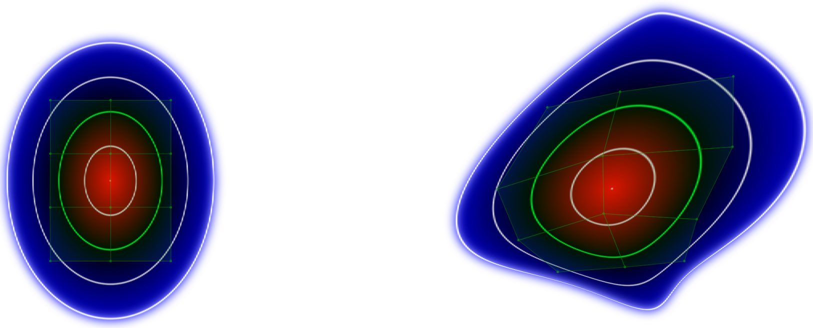


FIGURE 25 – **Résolution de la position inverse des *Free-Form Deformations***
Le tracé rouge représente les positions qui correspondent aux λ_p où p est l'indice de l'itération. La résolution commence des coordonnées λ_0 induites pas la position de contrôle la plus proche jusqu'à convergence vers la correspondance qui donne $\mathcal{B}(\lambda) = Q$. Ceci permet depuis l'espace d'origine de retrouver P tel que déformé donne Q : $\mathcal{B}(\mu(P)) = Q$.

Une fois la convergence atteinte en $\lambda = (s, t, u)$, la position inverse est retrouvée depuis l'évaluation de la position dans la cage d'origine :

$$P = \sum_{(i,j,k)\in[l]\times[m]\times[n]} \binom{l}{i}\binom{m}{j}\binom{n}{l} s^i (1-s)^{l-i} t^j (1-t)^{m-j} u^k (1-u)^{n-k} P_{i,j,k}.$$



(a) Champ d'origine et sa cage (b) Champ déformé et la cage déformée

FIGURE 26 – **Déformation d'un champ scalaire par une *Free-Form Deformation* (2D)**
Déformation d'une surface implicite (champ scalaire d'un cercle) par une Free-Form Deformation. Implémentation 2D en `Java` *sous Processing IDE.*

Le principe ici peut être vu comme l'idée de naviguer dans l'espace d'origine en se déplaçant à chaque pas vers la direction optimale qui permettra la correspondance avec la position souhaitée dans l'espace déformé.

7.4 Architecture de nos méthodes

Les champs scalaires déformés seront définis en 2D depuis un prototype codé en `Java` utilisant *Processing IDE* pour le rendu et en 3D par un prototype en `C++` utilisant *Radium Engine* (le moteur graphique proposé par l'équipe STORM). Les résultats expérimentaux ont pour but d'illustrer aussi bien les résultats de nos méthodes que leurs défauts.

Nos méthodes supposent pour le moment que le système de coordonnées barycentrique choisi propose aussi une interpolation à l'extérieur de notre cage puisque le champ scalaire n'est pas restreint à l'intérieur. Imposer que le champ scalaire soit englobé par la cage imposerait à un infographiste des contraintes qu'il n'a pas sans utiliser l'Implicit skinning sur le maillage ou la cage, ce qui n'est pas souhaité. Le système de coordonnées utilisé pour nos tests est donc les *Mean Value Coordinates* qui est le système de coordonnées le plus populaire qui défini aussi l'interpolation à l'extérieur de la cage. Ceci bien que notre architecture ne soit pas limitée à la seule utilisation des *Mean Value Coordinates*. Je propose en figure 27 l'architecture logicielle de nos méthodes d'évaluation d'un champ scalaire déformé.

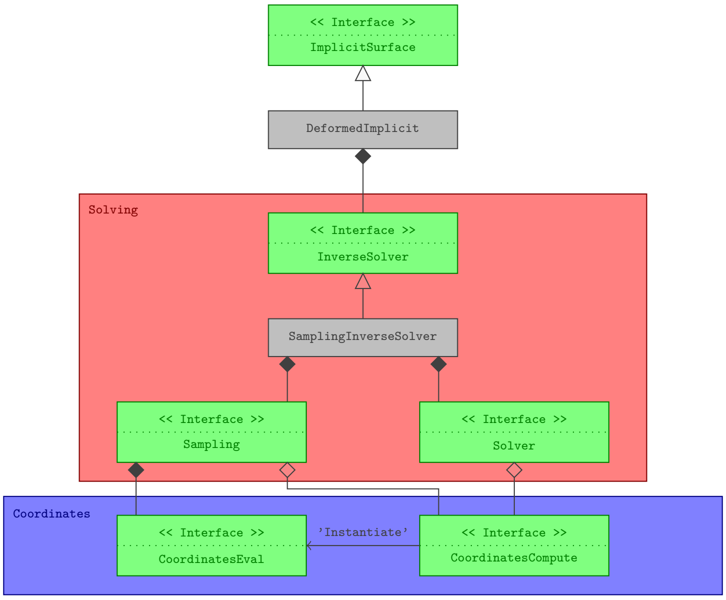


FIGURE 27 – **Architecture de base pour l'évaluation d'une surface implicite déformée**
Une surface implicite déformée `DeformedImplicit` *est évaluée comme une surface implicite classique* `ImplicitSurface`*. Ce qui permet son évaluation est l'outil de résolution qu'elle utiliser pour résoudre le problème de la position inverse* `InverseSolver`*. Cet outil se décompose en deux partie une première partie* `Sampling` *qui correspond à un échantillonnage de l'espace qui sera déformé par le système de coordonnées barycentriques et servira à la détection de positions dans l'espace déformé et une seconde partie* `Solver` *qui correspond à une méthode itérative de reconstruction de la position inverse.* `CoordinatesCompute` *correspond au système de coordonnées barycentriques choisi et effectue l'étape de Binding de celui-ci ce qui produit les poids* `CoordinatesEval` *en une position qui permettent de retrouver toute position dans une cage déformée par l'évaluation des coordonnées barycentriques (l'évaluation peut varier selon le système de coordonnées).* `Sampling` *fait appel à* `CoordinatesCompute` *uniquement pour le Binding de l'échantillonnage puis se met à jour lors des déformations de la cage à l'aide de* `CoordinatesEval`*.*

L'intérêt de cette architecture est de pouvoir modifier le système de coordonnées barycentrique sans modifier notre résolution que de changer les composants de notre résolution. Ceci permet en particulier de remplacer une composante pour une méthode de détection ou une méthode de résolution différente. De plus, depuis cette architecture, nous proposons plusieurs méthodes et comparaisons selon les composants branchés. Une première méthode s'inspire de la résolution numérique directe proposée pour résoudre le problème de la position inverse avec les *Free-Form Deformations*. La seconde méthode propose une détection qui permet l'approximation d'antécédents multiples et s'inspire de l'idée de guider la déformation d'une grille régulière par la déformation du système de coordonnées.

8 Première approche : résolution numérique directe

8.1 Principe de la méthode

La première approche proposée se base sur l'idée de résoudre numériquement la position inverse induite par la transformation comme proposé plus haut. Nous avons testé deux résolutions à ce sujet :

— Une première qui se base sur l'idée de passer de l'espace d'origine dans le repère des coordonnées cartésiennes vers l'espace déformé dans le repère des coordonnées cartésiennes. Ceci permet une résolution directe de la transformation d'une position vers une autre.

— Une seconde qui propose de naviguer dans l'espace de coordonnées barycentriques pour établir des coordonnées qui font la correspondance entre la position dans chaque espace (ce qui est fait avec les *Free-Form Deformations*).

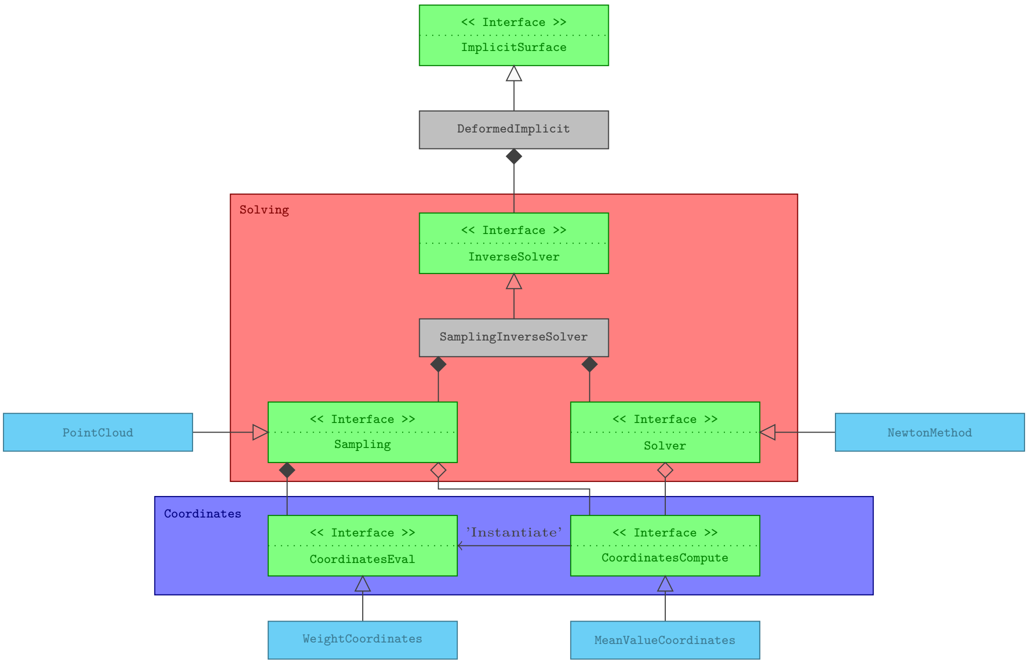


FIGURE 28 – **Architecture de la méthode de résolution numérique directe**
La méthode se base sur l'échantillonnage de l'espace par un nuage de points `PointCloud` *dont on calcule les coordonnées barycentriques généralisées Binding. De même que pour les Free-Form Deformations nous proposons d'utiliser la méthode de Newton pour résoudre le problème de la position inverse. Pour l'évaluation de la position inverse* `SamplingInverseSolver` *demandera au* `Sampling` *l'échantillon le plus proche puis résout le problème de la position inverse par le* `Solver`*.*

Cette méthode se base sur l'architecture en figure 28. On échantillonne l'intérieur de la cage (le système de coordonnées interpole toujours l'intérieur de la cage) : ici l'échantillonnage se fait utilisant des triangles/tétrahèdres liant chaque arête/face au centre de masse de la cage. Puis on les utilise pour obtenir les échantillons qui sont les positions issues de la subdivision des triangles/tétrahèdres par leurs coordonnées barycentriques (Pour une résolution $r \in \mathbb{N}$ et un triangle (ABC), on prend les positions $\left\{\frac{i}{r}A + \frac{j}{r}B + \frac{k}{r}C | (i,j,k) \in \mathbb{N}^3 / i+j+k = r\right\}$). L'ensemble des échantillons sera noté $\{S_i\}_{i \in [\#échantillons]}$.

8.2 Résolution dans l'espace des coordonnées cartésiennes

La résolution dans l'espace des coordonnées cartésiennes se base sur l'idée qu'une position P de l'espace d'origine est transformée en une position Q dans l'espace déformé. L'idée est donc de résoudre directement le problème de la position inverse $T(P) = Q$. Ceci demande donc de calculer :

$$J_T(P) = \left(\frac{\partial}{\partial x}T(P), \frac{\partial}{\partial y}T(P), \frac{\partial}{\partial z}T(P)\right)$$

pour utiliser la résolution de Newton. L'utilisation de la matrice Jacobienne telle quelle pose des problèmes de convergence aux bords de la cage ce que je justifie en annexe B.1. Nous remplacerons donc ici, les dérivées partielles par un calcul depuis les différences finies :

$$\frac{\partial}{\partial \delta}T(P) \leftarrow \frac{T\left(P + h\frac{\partial}{\partial \delta}\begin{pmatrix}x\\y\\z\end{pmatrix}\right) - T(P)}{h}$$

pour $\delta \in \{x, y, z\}$ la variable à dériver un h arbitraire (dans notre cas, j'utilise $h = 10^{-2}$).

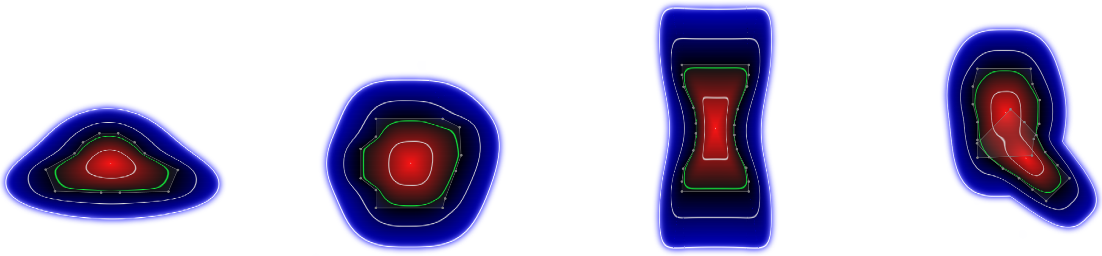
(a) Champ d'origine et sa cage (b) Champ déformé et sa cage

FIGURE 29 – **Déformation d'un champ scalaire par une cage pentagonale (2D)**
Déformation d'une surface implicite (champ scalaire d'un cercle) par une cage pentagonale et le système de coordonnées Mean Value Coordinates. Implémentation 2D en `Java` *sous Processing IDE.*

Pour une évaluation en une position Q dans l'espace déformé, nous recherchons l'échantillon déformé le plus proche $T\left(S_{i(Q)}\right)$. Nous connaissons sa correspondance dans l'espace d'origine d'où $P_0 = S_{i(Q)}$ la position de départ pour notre résolution. Ensuite les itérations seront données par :

$$P_{k+1} = P_k + J_T\left(P_k\right)^{-1}\left(T\left(P_k\right) - Q\right).$$

Cette résolution donne la déformation du champ en 2D proposée en figure 29. De même, je propose en figure 30 des exemples de déformations de la surface que peut vouloir effectuer un animateur. Cependant lors de déformations particulières qui engendrent des concavités de la cage, cette méthode est peu robuste et laisse apparaître des interférences (observé en annexe B.2).



(a) Aplatissement (b) Ajout de muscle (c) Étirement (d) Composition

FIGURE 30 – **Exemples de déformations de surfaces par une cage (2D)**
En (a) l'aplatissement d'une sphère pour simuler un effet physique. En (b) l'étirement de la surface pour simuler des muscles. En (c) l'étirement de la surface pour simuler un effet élastique. En (d) un example de composition de deux surfaces déformées par une cage à l'aide d'un opérateur de mixage.

Le principal défaut de cette méthode est le temps de calcul pour une évaluation. En effet, pour une évaluation dans la surface implicite d'origine nous somme de l'ordre de $160ns$, alors que l'évaluation par notre méthode est de l'ordre de $320\mu s$ (3 itérations) soit de l'ordre d'un facteur 2000 de différence. Pour comprendre le problème, pour l'évaluation d'une position par notre système de coordonnées nous somme de l'ordre de $180ns$. Mais pour effectuer le *binding* d'une position avec les *Mean Value Coordinates* ceci demande $10\mu s$. Cependant pour calculer la transformation T d'une position quelconque comme le fait notre méthode, ceci demande de passer par l'étape de *binding* puis l'évaluation de la transformation ce que j'ai mesuré à $11\mu s$. En effet, ici l'évaluation de la position est négligeable à côté du *binding* (de l'ordre d'un facteur 55).

Le problème est que pour calculer notre matrice Jacobienne ceci demande aussi de passer par l'étape de *binding* (ceci même en admettant de ne pas passer par les différences finies puisque ceci demanderait en plus de dériver notre système de coordonnées). Le calcul de la matrice Jacobienne par les différences finies demande 4 évaluations de la transformation ce que j'ai mesuré à $38\mu s$. De plus une itération demande une évaluation de la matrice Jacobienne et une évaluation de la transformation. Ceci nous fait de l'ordre de $5n$ évaluations de l'étape de *binding*. Pour 3 j'ai mesuré à $300\mu s$ le temps demandé pour la résolution. Le temps passé à cherche l'échantillon le plus proche est lui de l'ordre de $9\mu s$ ce qui reste négligeable à côté de la résolution. Ceci fait un temps de l'ordre de $320\mu s$ pour une évaluation du champ scalaire déformé.

À partir de là, l'idée des prochaines méthodes est donc de se libérer de l'étape de *binding* pendant la résolution. À noter que pour le cas des *Mean Value Coordinates* nous avons une forme close pour l'évaluation mais que d'autres système de coordonnées comme les *Harmonic Coordinates* demandent une résolution numérique pour le calcul des poids.

8.3 Résolution dans l'espace des coordonnées barycentriques

L'idée est donc de proposer la résolution numérique en naviguant dans le système de coordonnées barycentrique sans avoir à refaire l'étape de *binding* de la position. L'idée proposée est de résoudre le problème d'optimisation suivant :

$$\min_{\{\alpha_i\}_{i\in[n]}} \left\| \sum_{i\in[n]} \alpha_i T(C_i) - Q \right\|^2 .$$

Pour cela nous pouvons utiliser une descente de gradient. Chaque dérivée partielle en une coordonnée est donnée par :

$$\frac{\partial}{\partial \alpha_k} || \sum_{i\in[n]} \alpha_i T(C_i) - Q||^2 = 2T(C_k) \cdot \left(\sum_{i\in[n]} \alpha_i T(C_i) - Q \right) .$$

Ceci permet de calculer le gradient comme un produit matriciel :

$$\nabla \left\| \sum_{i\in[n]} \alpha_i T(C_i) - Q \right\|^2 = 2\ {}^t\left(T(C_1), T(C_2), \ldots, T(C_n)\right) \cdot \left(\sum_{i\in[n]} \alpha_i T(C_i) - Q \right) .$$

Depuis le sampling on prend $T\left(S_{i(Q)}\right)$ la plus proche position de Q. Puis que chaque position S_i a été bindé, il est possible de récupérer ses coordonnées barycentriques :

$$A\left(S_{i(Q)}\right) = \left(\alpha_1\left(S_{i(Q)}\right), \alpha_2\left(S_{i(Q)}\right), \ldots, \alpha_n\left(S_{i(Q)}\right)\right) .$$

Nous prendrons $A_0 = A\left(S_{i(Q)}\right)$ comme point de départ pour notre résolution. Puis chaque itération sera donnée par :

$$A_{k+1} = A_k - \left\| \sum_{i\in[n]} \alpha_i^k T(C_i) - Q \right\| \frac{\nabla \left\| \sum_{i\in[n]} \alpha_i^k T(C_i) - Q \right\|^2}{\left\| \nabla \left\| \sum_{i\in[n]} \alpha_i^k T(C_i) - Q \right\|^2 \right\|} .$$

La solution finale est donnée par A par suite, on retrouve une position inverse $P(Q) = \sum_{i\in[n]} \alpha_i^k C_i$. Mais, en réalité ceci ne permet pas de retrouver le champ attendu et la position inverse par le système de coordonnées comme illustré par les résultats en figure 31.

Dans le cas des *Free-Form Deformations*, ce type de déplacement dans l'espace des coordonnées ne posait pas de problème puisque les espaces en coordonnées cartésiennes et celui en coordonnées dans le volume de Bézier étaient de même dimension et la navigation respectait implicitement les contraintes imposées pour passer de l'un à l'autre. Dans le cas de notre descente de gradient, nous naviguons dans un espace plus vaste que l'espace des coordonnées cartésiennes. Notre gradient peut par conséquence

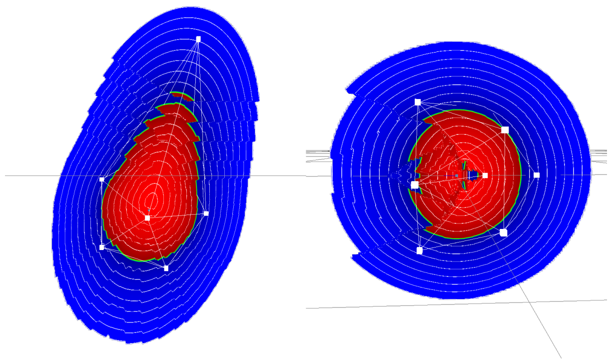


FIGURE 31 – **Résolution par une descente de gradient dans le système de coordonnées (3D)** *Vue en coupe de la résolution par la descente de gradient dans le système de coordonnées. On observe ici que le gradient du champ scalaire est continu localement dans des régions bien délimitées. Ces régions correspondent au choix de l'échantillon le plus proche.*

s'orienter vers toutes les directions de notre espace de coordonnées sans aucune garantie de rester des *Mean Value Coordinates* ou même simplement rester des coordonnées barycentriques.

Pour que cette méthode fonctionne, ceci demanderait de trouver une base orthogonale de contraintes $\{\mathfrak{C}_j(\{\alpha_i\})\}_{j\in[l]}$ définies par notre système de coordonnées puis de corriger le gradient par une projection orthogonale dans cet espace de contraintes :

$$\nabla_{\mathfrak{C}}\left\|\sum_{i\in[n]}\alpha_i T(C_i)-Q\right\|^2=\nabla\left\|\sum_{i\in[n]}\alpha_i T(C_i)-Q\right\|^2-\sum_{j\in[l]}\frac{\nabla\left\|\sum_{i\in[n]}\alpha_i T(C_i)-Q\right\|^2\cdot\mathfrak{C}_j(\{\alpha_i\})}{\mathfrak{C}_j(\{\alpha_i\})\cdot\mathfrak{C}_j(\{\alpha_i\})}\mathfrak{C}_j(\{\alpha_i\}).$$

Mais ceci rend bien plus difficile l'idée de rester générique à tout système de coordonnées barycentriques et de plus risque de revenir à des temps de calcul qui peuvent demander une expression des contraintes en fonction du *binding* de la position. Nous n'avons donc pas creusé cette piste.

Cette approche et les deux résolutions résolutions proposées sont ni satisfaisantes ni utilisables. Ceci nous a donc conduit à cherche à approximer le système de coordonnées au lieu de chercher une résolution directe et exacte.

9 Seconde approche : capture de la déformation pour approximation

9.1 Principe de la méthode

L'approche suivante se base sur l'idée de capturer la déformation du système de coordonnées à l'aide d'une grille régulière. C'est-à-dire que chaque position est bindée dans le système de coordonnées choisi (ceci suppose que le système de coordonnées définisse aussi une interpolation hors de la cage) comme illustré en figure 32. Puis l'idée est de détecter la cellule de la grille déformée dans laquelle se trouve la position que l'on cherche à évaluer pour ensuite reconstruire sa position depuis une interpolation trilinéaire dans la cellule qui possède par conséquent une correspondance dans la grille d'origine.

Cette méthode n'est pas exacte puisqu'elle se base une représentation discrète de notre déformation et approxime au mieux une solution. La méthode suit l'architecture donnée en figure 33. L'avantage de cette méthode est qu'elle ne demande à ce que le *binding* ne soit fait qu'une seule fois comme pour une déformation par cage classique.

9.2 Détection de correspondance

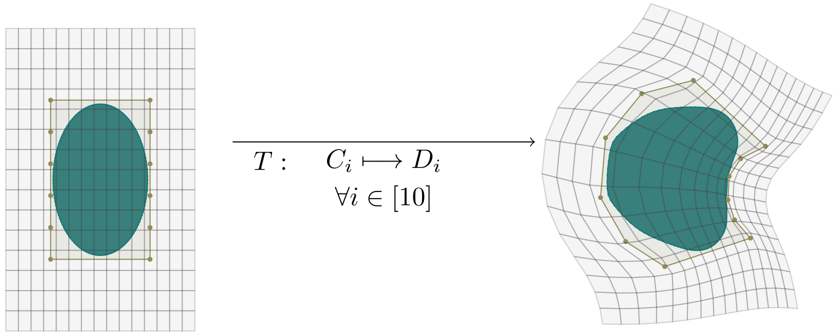


FIGURE 32 – **Capture de la déformation par une grille pour approximation**
L'interpolation donnée par le système de coordonnées est capturé par une grille régulière (qui englobe le support de la surface implicite). Lorsque la cage est déformée, la grille est déformée par le système de coordonnées, ceci permet d'avoir une échantillonnage discret de la déformation de l'espace.

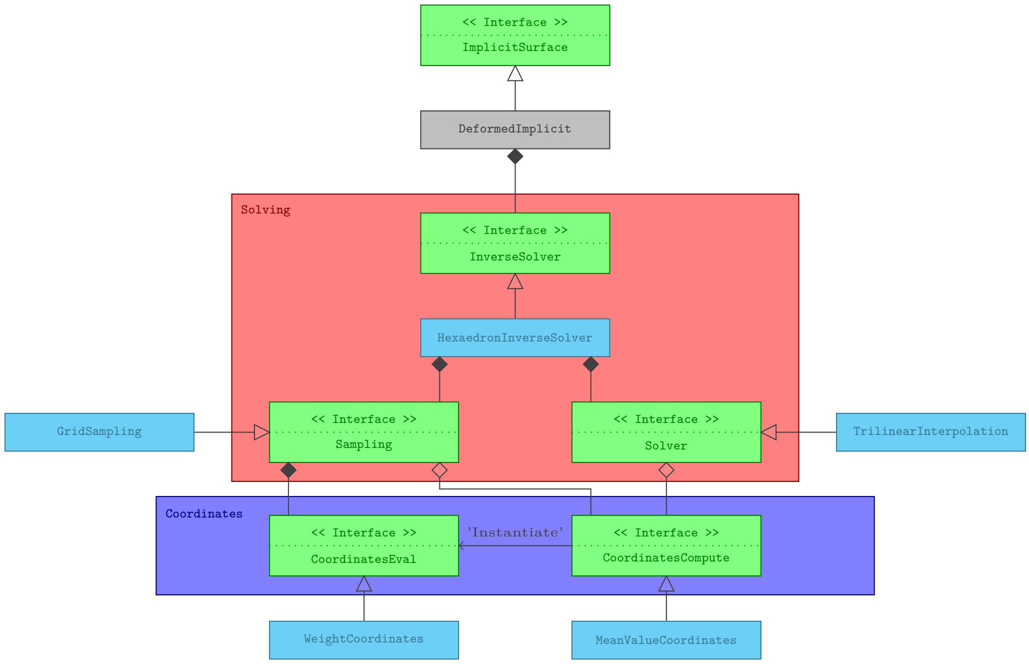


FIGURE 33 – **Architecture de la méthode de résolution par approximation dans une grille discrète**
L'architecture de la résolution par approximation est compatible avec l'architecture de base pour nos méthodes (établie en figure 27). Ici, `GridSampling` *est responsable de la détection de l'hexaèdre déformé (cellule) englobant le fetch (position où on demande l'évaluation du champ donc d'une position inverse). Ensuite,* `Solver` *sera en charge de reconstruire la position inverse par une approximation dans la grille d'origine depuis l'information donnée par* `GridSampling`*.* `HexahedronInverseSolver` *est responsable des requêtes qu'il fait au* `Sampling` *et passe les informations obtenues pour la reconstruction au* `Solver`*.*

9.2.1 Plus proche voisin et produit scalaire

Dans un premier temps nous avons gardé l'idée proposée qui consiste à sélectionner le plus proche voisin. Ici, pour des soucis de performances, nous utilisons un *Kd-tree*, c'est une structure de données permettant un partitionnement de l'espace. À chaque modification de notre cage, ceci entraîne une modification de notre grille par le système de coordonnées ce qui demande de reconstruire le *Kd-tree*.

Pour un fetch Q, la détection propose de sélectionner le sommet $Q_{i,j,k}$ de la grille le plus proche de Q. Puis de déterminer dans quelle hexaèdre se trouve Q en vérifiant si $\overrightarrow{Q_{i,j,k}Q}$ est de même orientation que la diagonale de l'hexaèdre par rapport aux faces qui intersectent $Q_{i,j,k}$. Ceci peut être vérifié rapidement

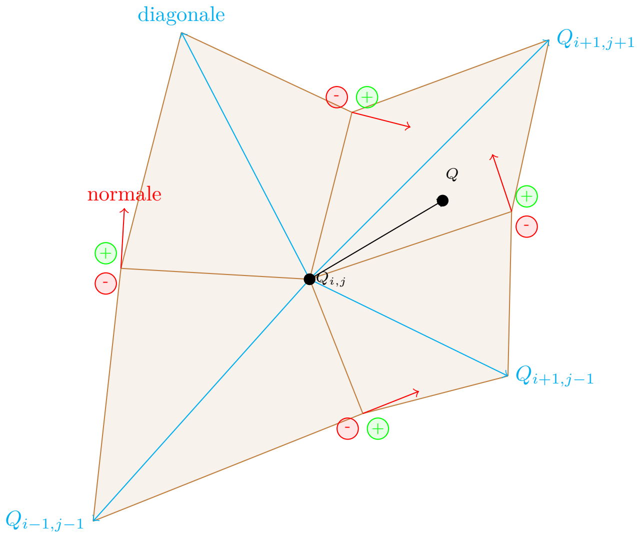


FIGURE 34 – **Détection par produit scalaire et vectoriel depuis le plus proche voisin (représentation 2D)**
Cette méthode se base sur l'idée qu'on puisse détecter l'hexaèdre englobant en vérifiant que le vecteur entre le plus proche voisin et le fetch est orienté de la même manière que la diagonale de l'hexaèdre par rapport aux différentes faces qui intersectent le plus proche voisin. Si tous les produits scalaires avec chaque normale est de même signe, alors les deux vecteurs ont la même orientation et le fetch est détecté comme étant dans l'hexaèdre.

à l'aide de produits vectoriels qui construisent la normale en chaque face et par vérification des signes des produits scalaires pour l'orientation comme illustré en figure 34. Ceci demande donc que pour un hexaèdre donné par l'offset $(o_x, o_y, o_z) \in \{-1, 1\}^3$, il vérifie la condition suivante :

$$\begin{aligned}
& \overrightarrow{Q_{i,j,k}Q} \cdot \left(\overrightarrow{Q_{i,j,k}Q_{i+o_x,j,k}} \wedge \overrightarrow{Q_{i,j,k}Q_{i,j+o_y,k}}\right) \times \overrightarrow{Q_{i,j,k}Q_{i+o_x,j+o_y,k+o_z}} \cdot \left(\overrightarrow{Q_{i,j,k}Q_{i+o_x,j,k}} \wedge \overrightarrow{Q_{i,j,k}Q_{i,j+o_y,k}}\right) \geqslant 0 \\
\wedge\ & \overrightarrow{Q_{i,j,k}Q} \cdot \left(\overrightarrow{Q_{i,j,k}Q_{i+o_x,j,k}} \wedge \overrightarrow{Q_{i,j,k}Q_{i,j,k+o_z}}\right) \times \overrightarrow{Q_{i,j,k}Q_{i+o_x,j+o_y,k+o_z}} \cdot \left(\overrightarrow{Q_{i,j,k}Q_{i+o_x,j,k}} \wedge \overrightarrow{Q_{i,j,k}Q_{i,j,k+o_z}}\right) \geqslant 0 \\
\wedge\ & \overrightarrow{Q_{i,j,k}Q} \cdot \left(\overrightarrow{Q_{i,j,k}Q_{i,j+o_y,k}} \wedge \overrightarrow{Q_{i,j,k}Q_{i,j,k+o_z}}\right) \times \overrightarrow{Q_{i,j,k}Q_{i+o_x,j+o_y,k+o_z}} \cdot \left(\overrightarrow{Q_{i,j,k}Q_{i,j+o_y,k}} \wedge \overrightarrow{Q_{i,j,k}Q_{i,j,k+o_z}}\right) \geqslant 0.
\end{aligned}$$

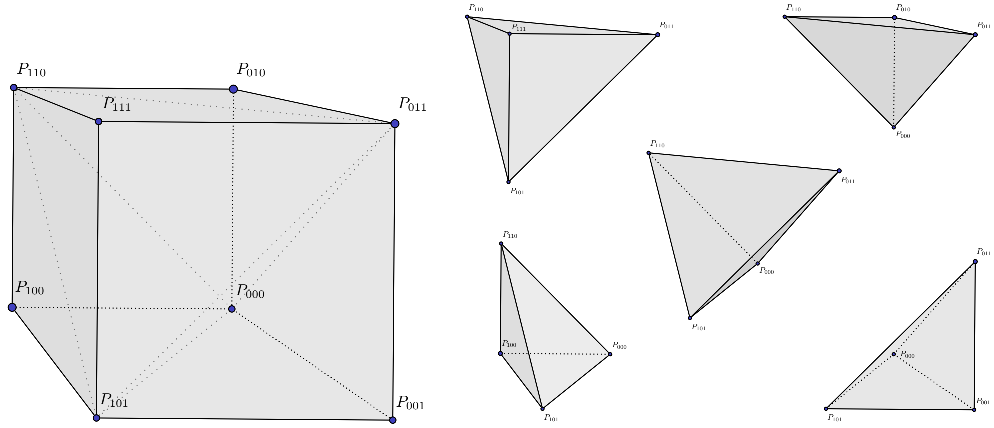
L'intérêt de cette première approche est qu'elle nous fait gagner de l'ordre d'un facteur 30 sur la précédente méthode. Mais, son principe de détection est en réalité peu robuste pour des déformations trop larges en particulier lorsque celles-ci entraînement une compression de l'espace ce qui est donné plus en détails en annexe C.1.

9.2.2 Détection basée tétraèdre

La solution proposée ici est donc de changer de méthode de détection pour une détection basée tétraèdre. En effet, la structure proposée est une grille régulière qui se déforme en un ensemble d'hexaèdres à faces non planes. Mais, il est possible de décomposer un hexaèdre en tétraèdres. La plus petite décomposition construit 5 tétraèdres depuis 1 hexaèdre (illustrée en figure 35). Depuis cette méthode il est possible de détecter si une position appartient à un tétraèdre à l'aide des coordonnées barycentriques par un produit matrice vecteur comme expliqué dans l'annexe A.1 sur les coordonnées barycentriques.

Pour une détection plus efficace nous utilisons une hiérarchie d'intervalles englobants pour partitionner l'espace et ordonner nos tétraèdres. Pour chaque tétraèdre nous calculons sa matrice inverse pour la détection : $\left(\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{AD}\right)^{-1}$. Pour une position P, il suffit alors de calculer $(s,t,u) = \left(\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{AD}\right)^{-1} \overrightarrow{AP}$ puis vérifier que

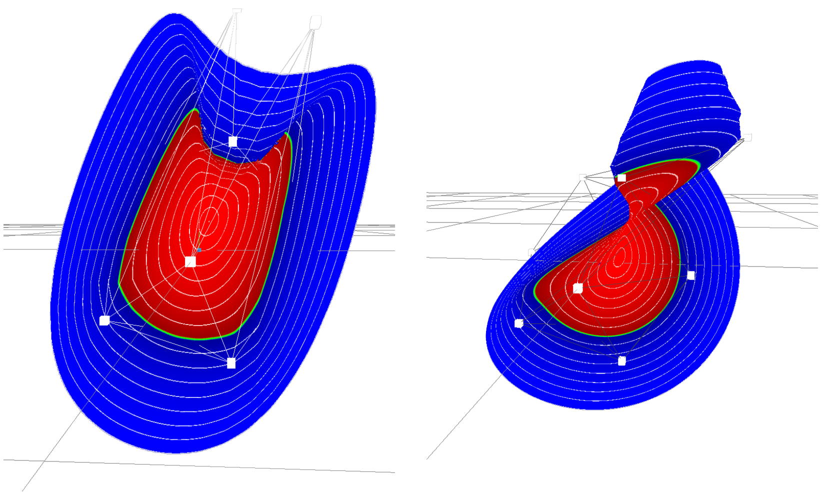
$$\begin{cases} \min\left(s,t,u,s+t+u\right) \geqslant 0 \\ \max\left(s,t,u,s+t+u\right) \leqslant 1. \end{cases}$$



(a) Hexaèdre (b) Décomposition en tétraèdres

FIGURE 35 – **Décomposition d'un hexaèdre en tétraèdres**

L'intérêt de cette méthode est à la fois sa robustesse aux déformations d'un hexaèdre, mais aussi la possibilité de détecter plusieurs antécédents (ceci sera vu plus en détail dans la section 9.3). Dans un premier temps, il est possible de proposer une solution dont l'idée est de faire un max sur les valeurs scalaires des différents antécédents ce qui permet une résolution plus propre de notre champ comme illustré en figure 36.



(a) Large concavité (b) Repliement de la cage

FIGURE 36 – **Démonstration de la robustesse de la détection basée tétraèdre**
Cette figure illustre la résolution du problème observé en annexe C.1. En (a) est illustré la résolution du problème d'interférence excessive observé précédemment. En (b), on observe que la détection reste robuste même pour des déformations qui ne font plus sens en animation.

9.3 Détection d'antécédents multiples

Le principal intérêt de la méthode proposée est qu'elle permet la résolution d'antécédents multiples. C'est-à-dire que puisque la déformation de l'espace est surjective, il se peut qu'il existe plusieurs antécédents pour une même image. En particulier ceci arrive lorsque la surface s'auto-intersecte comme illustré en figure 37.

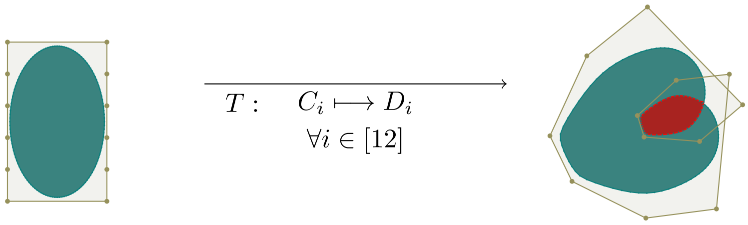


FIGURE 37 – **Exemple d'auto-intersection**
Déformation de la cage qui entraîne une auto-intersection de la surface (en rouge).

Une détection basée plus proche voisin ne peut pas détecter des cas d'antécédents multiples et est mise en défaut lorsque ce type de cas arrive. Mais la méthode basée tétraèdres fonctionne du fait qu'elle s'initialise dans un espace où les tétraèdres sont deux-à-deux disjoints. Ceci permet lorsqu'une position appartient à plusieurs tétraèdres de détecter un cas d'antécédents multiples. Par suite, nous connaissons les positions initiales des tétraèdres ce qui permet de résoudre leur position inverse pour chaque tétraèdre. En effet, le fait de commencer suffisamment proche d'un zéro de notre fonction $T(P) - Q$ permet de capturer localement les différents antécédents comme illustré en figure 38.

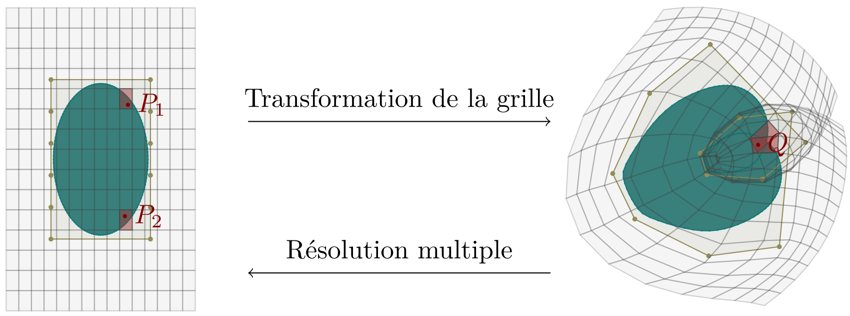


FIGURE 38 – **Détection multiple et auto-intersection**
La position Q est détectée dans deux hexaèdres issus la grille déformée. Ces hexaèdres donnent une correspondance avec la grille d'origine. Ceci permet la reconstruction des antécédents P_1 et P_2 dans l'espace d'origine tels que $T(P_1) = T(P_2) = Q$.

9.4 Reconstruction du champ scalaire

La reconstruction proposée pour le champ scalaire est une reconstruction trilinéaire dans chaque hexaèdre. Ici, nous n'avons pas de propriété de régularité ou d'hexaèdres à faces planes. Je propose donc de résoudre le problème de la position inverse dans un hexaèdre en utilisant la méthode de Newton sur le même principe que les résolutions proposées précédemment.

La reconstruction trilinéaire dans un hexaèdre $\{D_{ijk}\}_{i,j,k\in\{0,1\}^3}$ se base sur l'idée de trouver des coordonnées s, t, u telles que :

$$Q = \sum_{i,j,k\in\{0,1\}^3} s^i (1-s)^{1-i} t^j (1-t)^{1-j} u^k (1-u)^{1-k} D_{ijk}.$$

Ceci s'écrit aussi :

$$Q = V_{000} + sV_{001} + tV_{010} + uV_{100} + stV_{011} + suV_{101} + tuV_{110} + stuV_{111}$$

en posant :

$$
\begin{array}{l}
V_{000} = D_{000}, V_{001} = (D_{001} - D_{000}), V_{010} = (D_{010} - D_{000}), V_{100} = (D_{100} - D_{000}) \\
V_{011} = (D_{000} + D_{011} - D_{010} - D_{001}) \\
V_{101} = (D_{000} + D_{101} - D_{100} - D_{001}) \\
V_{110} = (D_{000} + D_{110} - D_{100} - D_{010}) \\
V_{111} = (D_{111} + D_{100} + D_{010} + D_{001} - D_{011} - D_{101} - D_{110} - D_{000}).
\end{array}
$$

Par suite la jacobienne de la transformation affine $T : \{C_{ijk}\}_{i,j,k\in\{0,1\}^3} \mapsto \{D_{ijk}\}_{i,j,k\in\{0,1\}^3}$ est donnée par les dérivées partielles :

$$
\begin{array}{l}
\frac{\partial}{\partial s} T(s,t,u) = V_{001} + tV_{011} + uV_{101} + tuV_{111}, \\
\frac{\partial}{\partial t} T(s,t,u) = V_{010} + sV_{011} + uV_{110} + suV_{111}, \\
\frac{\partial}{\partial u} T(s,t,u) = V_{100} + sV_{101} + tV_{110} + stV_{111}.
\end{array}
$$

Dans un premier temps la reconstruction proposait d'utiliser la méthode de Newton en choisissant comme position de départ l'un des sommets de l'hexaèdre. Mais en réalité, une idée plus pertinente est celle d'utiliser les tétraèdres pour obtenir la position de départ. C'est-à-dire que lors de la détection nous calculons les coordonnées barycentriques dans notre tétraèdre, il est donc possible de calculer une position depuis ces coordonnées dans le tétraèdre de la grille d'origine avec un coût négligeable.

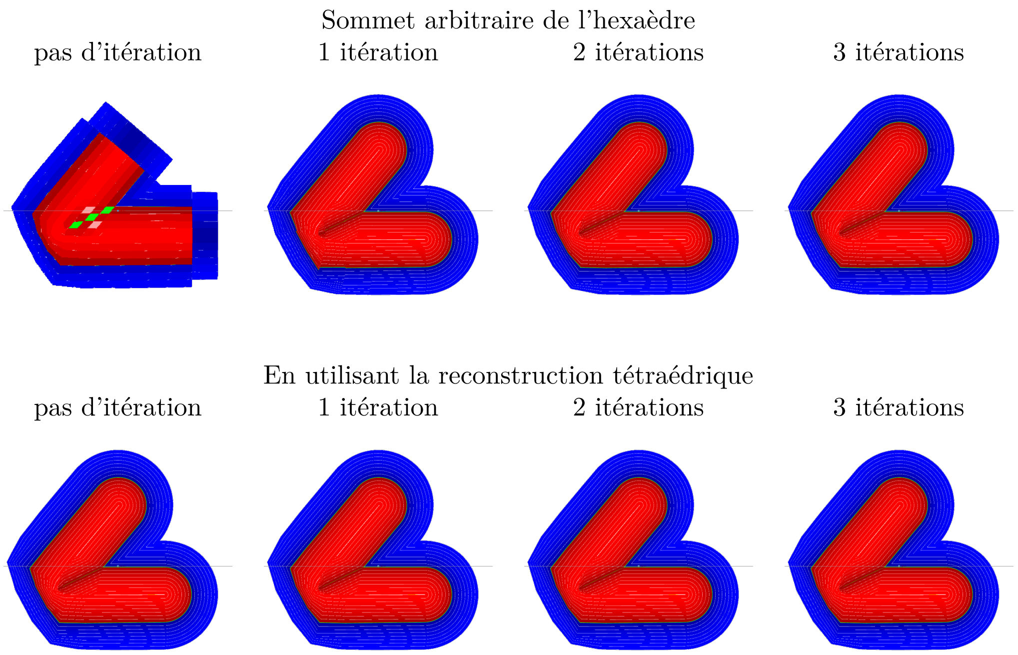
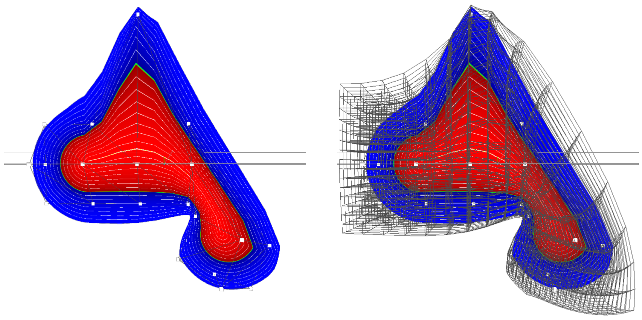


FIGURE 39 – **Itérations selon position de départ (sommet d'un hexaèdre et reconstruction tétraédrique)**
Résultat du champ scalaire reconstruit en fonction du nombre d'itérations effectuées par la méthode de Newton. En premier temps en utilisant un sommet arbitraire d'un hexaèdre, puis le résultat en proposant en d'initialiser la reconstruction depuis une reconstruction tétraédrique.

De plus nous considérons qu'un antécédent appartient à un tétraèdre lors de la détection et que l'hexaèdre est uniquement utilisé pour reconstruire une approximation de l'antécédent. Ceci permet d'être plus robuste lors que plusieurs tétraèdres de la décomposition d'un hexaèdre s'intersectent. De plus ceci donne de meilleurs résultats demandant moins d'itérations à la méthode de Newton (figure 39) et la reconstruction tétraédrique donne une meilleure position de départ pour converger vers l'antécédent local. À noter que dans le cas de la reconstruction trilinéaire elles sont presque identiques puisque les coordonnées barycentriques dans un tétraèdre proposent une interpolation linéaire. Mais la reconstruction tétraédrique reste pertinente pour d'autres reconstructions.

Le problème posé par la reconstruction trilinéaire est la non-continuité du gradient au bord des hexaèdres (figure 40). En effet, ceci est bien visible sur l'exemple de la capsule car les lignes droites sont préservées. ce qui n'est pas la déformation voulue. Ceci est dû à la linéarité de l'interpolation dans les cellules. Pour régler ce problème nous proposons d'utiliser une reconstruction donnant une interpolation plus lisse, par exemple en utilisant les bases de fonctions splines.



(a) Champ (b) Champ avec sa grille

FIGURE 40 – **Problème de continuité du gradient aux bords de hexaèdres**
La déformation du champ (a) préserve les lignes droites de la capsule dans chaque hexaèdre (b) ce qui entraîne des discontinuités du gradient aux bords des hexaèdres.

La reconstruction par une B-spline d'ordre p dans une grille de résolution $n \times n \times n$ s'écrit :

$$Q = \sum_{i,j,k \in [n]^3} \mathcal{N}_i^p(s)\, \mathcal{N}_j^p(t)\, \mathcal{N}_k^p(u)\, \mathfrak{Q}_{i,j,k}$$

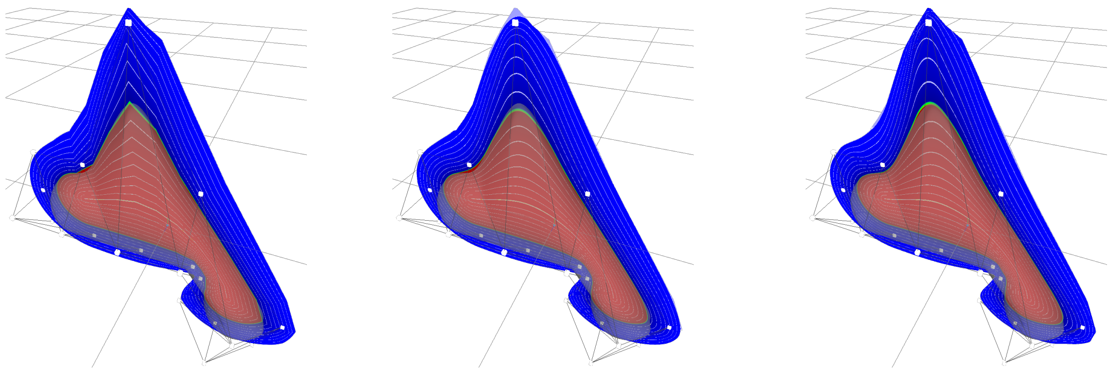
où (s, t, u) sont les coordonnées de Q dans la grille formée des positions de contrôle $\{\mathfrak{Q}_{i,j,k}\}_{i,j,k \in [n]^3}$ et $\{\mathcal{N}_i^p\}_{i \in [n]}$ la base de fonctions spline d'ordre p. Une base de fonction spline d'ordre p offre une continuité $\mathcal{C}^{p-2}$. Chaque fonction de cette base est un polynôme par morceaux de degré $p-1$. L'interpolation trilinéaire correspond à une spline d'ordre 2 et n'offre une continuité que $\mathcal{C}^0$ d'où la discontinuité du gradient. Pour avoir un gradient continu, nous proposons une reconstruction triquadratique depuis une B-spline d'ordre 3.

Une spline s'appuie sur la définition d'un vecteur nodal de $n + k + 1$ valeurs scalaires. Ce vecteur nodal permet de définir une coordonnée de notre position sur une courbe définie depuis nos positions de contrôle. Dans notre cas, nous proposons de conserver des coordonnées dans $[0, 1]$, d'où le vecteur nodal exprimé par les scalaires suivants :

$$u_i = \frac{i - (k-1)}{n - (k-1)}.$$

L'idée pour reconstruire un volume depuis une B-spline comme nous le proposons est de partir de la base de fonctions $\{\mathcal{N}_i^p\}_{i \in [n]}$ définie pour interpoler une courbe. Puis on ajoute une dimension en combinant deux courbes splines par le produit tensoriel ce qui revient à interpoler une surface depuis deux coordonnées (s, t) depuis la base de fonctions $\left\{\mathcal{N}_i^p(s)\, \mathcal{N}_j^p(t)\right\}_{i,j \in [n]^2}$. Dans la même idée en utilisant à nouveau le produit tensoriel ceci nous donne une base de fonctions trivariées pour interpoler un volume comme exprimé plus haut.
Il est possible de construire une base de fonction spline depuis une définition récursive de celle-ci :

$$\begin{cases} \mathcal{N}_i^1(u) = \mathbb{1}_{[u_i, u_{i+1}[}(u) \\ \mathcal{N}_i^p(u) = \frac{u - u_i}{u_{i+k-1} - u_i} \mathcal{N}_i^{p-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} \mathcal{N}_{i+1}^{p-1}(u)\,. \end{cases}$$



(a) Interpolation trilinéaire (b) Approximation triquadratique (c) Interpolation triquadratique

FIGURE 41 – **Amélioration de la continuité du gradient par les B-splines**
Le problème de continuité du gradient (a) est résolu par (b) et (c). Les B-splines étant approximantes (b) par défaut ceci demande de résoudre un système linéaire pour les rendre interpolantes (c).

Cependant, si nous supposons qu'il est possible de définir les points de contrôle de la B-spline comme les positions de la grille déformée, la B-spline se comporterait comme un approximant. En effet, pour utiliser une B-spline comme un interpolant et donc obtenir notre interpolation triquadratique, ceci demande à ce qu'elle vérifie les équations suivantes :

$$\sum_{i,j,k\in[n]^3} \mathcal{N}_i^p\left(\frac{\alpha-1}{n-1}\right)\mathcal{N}_j^p\left(\frac{\beta-1}{n-1}\right)\mathcal{N}_k^p\left(\frac{\gamma-1}{n-1}\right)\mathfrak{Q}_{i,j,k} = T\left(P_{\alpha,\beta,\gamma}\right)$$

avec $(\alpha,\beta,\gamma)\in[n]^3$ et $T\left(P_{\alpha,\beta,\gamma}\right)$ les positions de la grille déformée. Ceci impose que pour les coordonnées $(s,t,u)=\left(\frac{\alpha-1}{n-1},\frac{\beta-1}{n-1},\frac{\gamma-1}{n-1}\right)$ la spline passe en la position $T\left(P_{\alpha,\beta,\gamma}\right)$. Dans le cas d'une courbe il est simple de la poser comme un système linéaire :

$$\begin{cases} \sum_{i\in[n]} \mathcal{N}_i^p\left(\frac{1-1}{n-1}\right)\mathfrak{Q}_i = Q_1 \\ \vdots \\ \sum_{i\in[n]} \mathcal{N}_i^p\left(\frac{\alpha-1}{n-1}\right)\mathfrak{Q}_i = Q_\alpha \\ \vdots \\ \sum_{i\in[n]} \mathcal{N}_i^p\left(\frac{n-1}{n-1}\right)\mathfrak{Q}_i = Q_n \end{cases}$$

puis un produit matriciel :

$$\mathcal{M}_n^p\cdot\mathfrak{Q} = \begin{pmatrix} \mathcal{N}_1^p\left(\frac{1-1}{n-1}\right) & \mathcal{N}_2^p\left(\frac{1-1}{n-1}\right) & \cdots & \cdots & \mathcal{N}_n^p\left(\frac{1-1}{n-1}\right) \\ \mathcal{N}_1^p\left(\frac{2-1}{n-1}\right) & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \mathcal{N}_i^p\left(\frac{\alpha-1}{n-1}\right) & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \mathcal{N}_n^p\left(\frac{(n-1)-1}{n-1}\right) \\ \mathcal{N}_1^p\left(\frac{n-1}{n-1}\right) & \cdots & \cdots & \mathcal{N}_{n-1}^p\left(\frac{n-1}{n-1}\right) & \mathcal{N}_n^p\left(\frac{n-1}{n-1}\right) \end{pmatrix}\cdot\begin{pmatrix}\mathfrak{Q}_1\\ \vdots \\ \mathfrak{Q}_i \\ \vdots \\ \mathfrak{Q}_n\end{pmatrix} = \begin{pmatrix}Q_1\\ \vdots \\ Q_\alpha \\ \vdots \\ Q_n\end{pmatrix}.$$

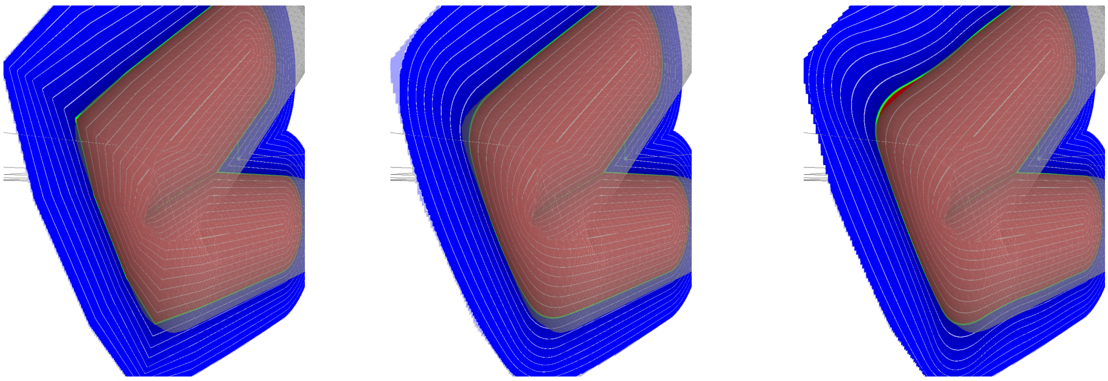
Par suite, la B-spline volumique a été fabriquée depuis le produit tensoriel des bases de fonctions d'une courbe spline, ce qui nous permet d'exprimer le système linéaire en trivarié par l'inversion de la matrice donnée par le produit tensoriel des matrices $\mathcal{M}_n^p$:

$$\left(\mathcal{M}_n^p\otimes\mathcal{M}_n^p\otimes\mathcal{M}_n^p\right)^{-1}.$$

Cette matrice n'a besoin d'être calculée qu'une seule fois pour p et n fixés. Je propose de l'inverser en tant que pré-calcul et la sauvegarder dans un fichier. Ceci demande alors simplement son chargement pour calculer les positions de contrôle $\{\mathfrak{Q}_{i,j,k}\}_{i,j,k\in[n]^3}$:

$$(\mathcal{M}_n^p \otimes \mathcal{M}_n^p \otimes \mathcal{M}_n^p)^{-1} \begin{pmatrix} Q_{1,1,1} \\ Q_{1,1,2} \\ \vdots \\ Q_{\alpha,\beta,\gamma} \\ \vdots \\ Q_{n,n,n} \end{pmatrix} = \begin{pmatrix} \mathfrak{Q}_{1,1,1} \\ \mathfrak{Q}_{1,1,2} \\ \vdots \\ \mathfrak{Q}_{i,j,k} \\ \vdots \\ \mathfrak{Q}_{n,n,n} \end{pmatrix}.$$

Bien que cette reconstruction offre un champ déformé plus lisse (figure 41). Ceci peut aussi produire des grossissements lors de déformations coudées pour interpoler et conserver un plus grand ordre de continuité (figure 42). Pour limiter ce type d'effets, ceci demande prendre une résolution de grille plus importante. Le problème d'augmenter la résolution est l'impact sur les temps de calcul (présenté en section 10.3) en particulier le temps de mise à jour de notre solveur.



(a) Interpolation trilinéaire (b) Approximation triquadratique (c) Interpolation triquadratique

FIGURE 42 – **Grossissement dû à l'augmentation de l'ordre de continuité**
En forçant la B-spline a être interpolante (c) ceci provoque des grossissements lors de déformations coudées dû au fait de devoir passer par toutes les positions et conserver l'ordre de continuité.

De plus, augmenter l'ordre de continuité provoquera des oscillations pour conserver un plus grand ordre de continuité jusqu'à être capable d'observer le phénomène de Runge avec l'interpolation par un polynôme de Lagrange multivarié (qui est $\mathcal{C}^\infty$).

10 Comparaison des approches et benchmark

Les benchmarks proposés sont effectués en CPU sur une machine 64 bits, avec un processeur Intel(R) Core(TM) i7 de fréquence 2.80GHz et 8Go de RAM.

10.1 Vue d'ensemble des temps de calcul

En figure 43, nous comparons le temps de calculs et le résultat de la déformation du champ scalaire d'une sphère de différentes méthodes proposées plus haut. La première méthode proposée effectue l'étape de *binding* pour la résolution d'une position pour une correspondance entre les deux espaces cartésiens ce à quoi elle doit son temps de rendu trop important. En tant que référence, il est aussi intéressant de comparer les temps obtenus par la méthode qui effectue la résolution numérique dans l'espace des coordonnées barycentriques. En effet, cette méthode demande un temps de calcul qui reste plus important que les méthodes d'approximation par une grille (de l'ordre d'au moins un facteur 2). Ceci fait qu'il semble plus intéressant de pousser les méthodes d'approximation que s'intéresser à la recherche des contraintes.

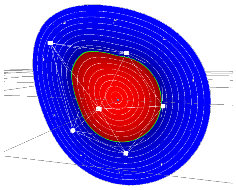
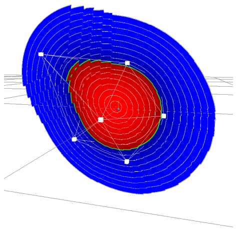
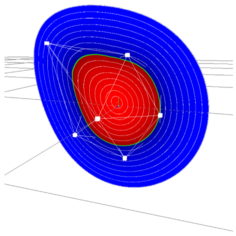
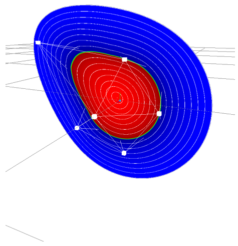
Méthode :	Résolution numérique		Approximation par grille (trilinéaire)	
	(Coordonnées cartésiennes)	(Coordonnées barycentriques)	(Détection plus proche voisin)	(Détection tétraèdres)
Résultat :				
Temps :	$3.82 \cdot 10^{10} ns (38.2s)$	$2.4 \cdot 10^{9} ns (2.4s)$	$1.29 \cdot 10^{9} ns (1.29s)$	$8.05 \cdot 10^{8} ns (805ms)$
Par fetch :	$1.45 \cdot 10^{5} ns (147\mu s)$	$9.16 \cdot 10^{3} ns (9.16\mu s)$	$4.92 \cdot 10^{3} ns (4.92\mu s)$	$3.07 \cdot 10^{3} ns (3.07\mu s)$

FIGURE 43 – **Benchmark des différentes approches sur une sphère**
Sphère en champ de distance borné déformée par une cage donnée par un géodésique : 12 sommets et 20 faces. Rendu par un plan de coupe de résolution 512×512 *soit* 2^{18} *évaluations.*

Concernant les méthodes d'approximation, elles permettent de gagner de l'ordre d'un facteur 30 par rapport à la méthode de résolution numérique (dans le cas des *Mean Value Coordinates*). De plus, un fait intéressant est que la méthode basée tétraèdres est plus efficace que la méthode basée plus proche voisin (en moyenne de l'ordre d'un facteur 1.3 sur les tests effectués). En effet, pour la méthode basée tétraèdre, ceci demande d'effectuer l'inversion d'une matrice pour calculer les coordonnées barycentriques mais une seule fois par tétraèdre. Ici, le produit matrice vecteur est plus rapide que le calcul d'une distance ce qui en fait aussi une méthode à la fois plus robuste et plus intéressante en temps de calcul.

10.2 Vue d'ensemble d'auto-intersections

En figure 44 nous comparons l'intérêt des différentes méthodes lorsque le phénomène des antécédents multiples se produit : ici lors d'une auto-intersection. La première méthode donne principalement des valeurs issues de l'intérieur de la capsule puisque son échantillonnage est définit à l'intérieur de la cage et donc donne des positions de départ qui en général convergent vers un antécédent à l'intérieur de la cage d'origine. La méthode de détection par le plus proche voisin semble très problématique ici. En effet, c'est le type de déformation qui va déformer la grille de sorte d'écraser et renverser des arêtes et donc qui met en défaut la détection de cette méthode. À titre de comparaison est proposé de sélectionner un seul antécédent issu du plus proche tétraèdre. Puis est donné le résultat qui consiste à prendre tous les antécédents trouvés et faire le max de leurs valeurs scalaires. Ceci donne un champ plus propre, dans l'idée de ce qu'on peut attendre pour ce type de déformation.

Dans la même idée, on illustre en figure 45 la résolution d'une mise en contact de la capsule avec elle-même. De même qu'en figure 44, nous observons les mêmes avantages et défauts des différentes méthodes. À noter que le temps passé pour la résolution d'antécédents multiples est un peu plus élevée que pour la sélection d'un antécédent car ceci demande la reconstruction de la position inverse pour chaque cellule détectée.

10.3 Méthodes de reconstruction pour approximation

Le paramétrage de la résolution de la grille (en figure 46) a principalement un impact sur le temps de mise-à-jour du solveur. En effet, le temps de mise-à-jour correspond au temps que le solveur prend pour reconstruire sa hiérarchie d'intervalles englobants pour ordonner les tétraèdres de la grille déformée. Dans le cas de l'interpolation triquadratique, ceci comprend aussi d'actualiser les positions de contrôle de la B-spline, ce qui est reste acceptable pour des petites résolutions mais peut très vite exploser et impacter le temps de mise-à-jour. Cependant, on peut remarquer que la résolution de la grille a peu d'impact sur le temps de fetch (implicitement donné par les temps des rendus). En effet, le coût payé

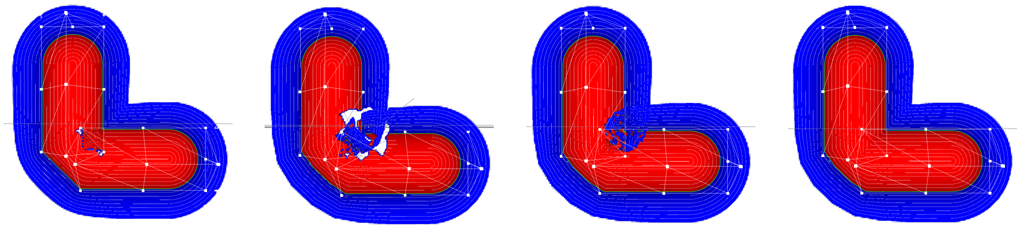
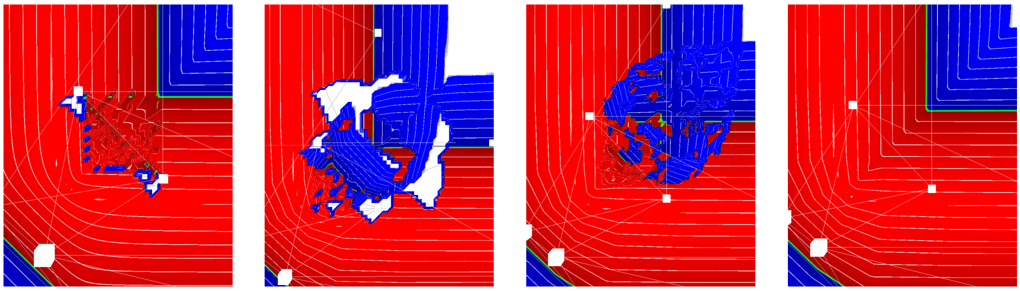
Méthode :	Résolution numérique (Coordonnées cartésiennes)	Approximation par grille (trilinéaire) (Détection plus proche voisin)	Approximation par grille (trilinéaire) (Détection plus proche tétraèdre)	Approximation par grille (trilinéaire) (Détection multiple)
Résultat :				
Zoom :				
Temps :	$1 \cdot 10^{11} ns (100s)$	$1.4 \cdot 10^{9} ns (1.4s)$	$6.93 \cdot 10^{8} ns (693ms)$	$7.17 \cdot 10^{8} ns (717ms)$
Par fetch :	$3.81 \cdot 10^{5} ns (381 \mu s)$	$5.34 \cdot 10^{3} ns (5.34 \mu s)$	$2.64 \cdot 10^{3} ns (2.64 \mu s)$	$2.74 \cdot 10^{3} ns (2.74 \mu s)$

FIGURE 44 – **Benchmark des différentes approches sur un pli**
Champ de distance borné d'une capsule (distance à un segment). Déformée par une cage de 26 sommets et 48 faces. Rendu par un plan de coupe de résolution 512×512 *soit* 2^{18} *évaluations.*

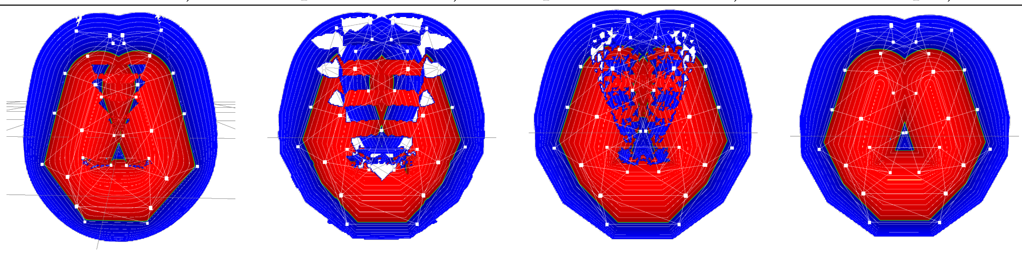
Méthode :	Résolution numérique (Coordonnées cartésiennes)	Approximation par grille (trilinéaire) (Détection plus proche voisin)	Approximation par grille (trilinéaire) (Détection plus proche tétraèdre)	Approximation par grille (trilinéaire) (Détection multiple)
Résultat :				
Temps :	$1.58 \cdot 10^{11} ns (158s)$	$2.2 \cdot 10^{9} ns (2.2s)$	$1.67 \cdot 10^{9} ns (1.67s)$	$1.75 \cdot 10^{9} ns (1.75s)$
Par fetch :	$6.02 \cdot 10^{5} ns (602 \mu s)$	$8.39 \cdot 10^{3} ns (8.39 \mu s)$	$6.37 \cdot 10^{3} ns (6.37 \mu s)$	$6.68 \cdot 10^{3} ns (6.68 \mu s)$

FIGURE 45 – **Benchmark des différentes approches sur un contact**
Champ de distance borné d'une capsule. Déformée par une cage de 42 sommets et 80 faces. Rendu par un plan de coupe de résolution 512×512 *soit* 2^{18} *évaluations.*

pour construire la structure de donnée pour la recherche de tétraèdres permet de ne payer que le coût de reconstruction lors d'un fetch puisqu'il devient négligeable. De plus, bien que la résolution du système linéaire soit chronophage, le chargement de la matrice une fois le pré-calcul effectué reste acceptable à l'initialisation. Ces problèmes de temps font qu'en général pour garder un aspect relativement temps réel lors de la déformation d'une cage, je propose de paramétrer la grille en résolution $16 \times 16 \times 16$.

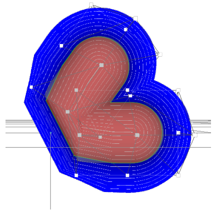
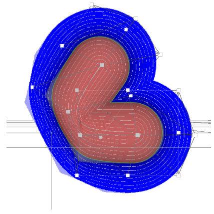
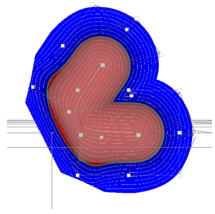
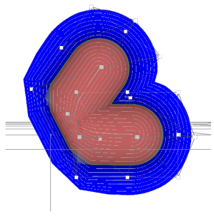
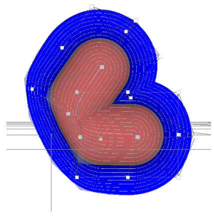
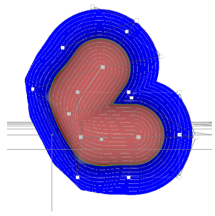
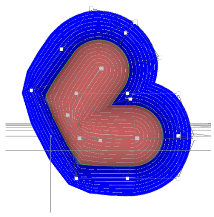
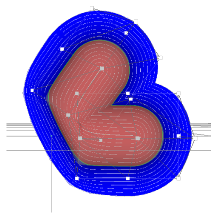
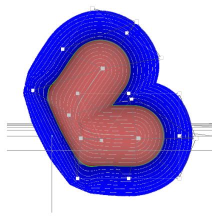
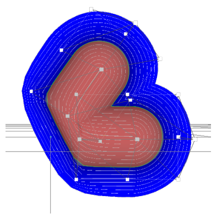
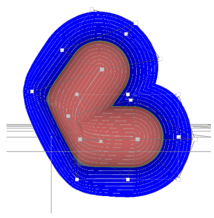
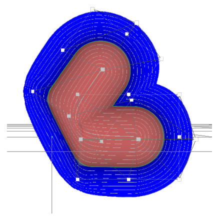
Reconstruction	Trilinéaire (Interpolation)	Triquadratique (Approximation)	Triquadratique (Interpolation)
Grille $8 \times 8 \times 8$			
Rendu	 $4.76 \cdot 10^8 ns(476ms)$	 $1.39 \cdot 10^9 ns(1.39s)$	 $1.31 \cdot 10^9 ns(1.31s)$
Résolution système			$1.23 \cdot 10^8 ns(123ms)$
Chargement système			$1.58 \cdot 10^6 ns(1.58ms)$
Mise à jour	$2.29 \cdot 10^6 ns(2.29ms)$	$2.63 \cdot 10^6 ns(2.63ms)$	$3.16 \cdot 10^6 ns(3.16ms)$
Grille $12 \times 12 \times 12$			
Rendu	 $4.34 \cdot 10^8 ns(434ms)$	 $1.29 \cdot 10^9 ns(1.29s)$	 $1.41 \cdot 10^9 ns(1.41s)$
Résolution système			$7.35 \cdot 10^9 ns(7.35s)$
Chargement système			$1.6 \cdot 10^7 ns(16ms)$
Mise à jour	$1.24 \cdot 10^7 ns(12.4ms)$	$1.11 \cdot 10^7 ns(11.1ms)$	$1.39 \cdot 10^7 ns(13.9ms)$
Grille $16 \times 16 \times 16$			
Rendu	 $4.67 \cdot 10^8 ns(467ms)$	 $1.3 \cdot 10^9 ns(1.3s)$	 $1.34 \cdot 10^9 ns(1.34s)$
Résolution système			$1.07 \cdot 10^{11} ns(1min47)$
Chargement système			$2.54 \cdot 10^8 ns(254ms)$
Mise à jour	$3.34 \cdot 10^7 ns(33.4ms)$	$3.43 \cdot 10^7 ns(34.3ms)$	$6.45 \cdot 10^7 ns(64.5ms)$
Grille $24 \times 24 \times 24$			
Rendu	 $4.97 \cdot 10^8 ns(497ms)$	 $1.39 \cdot 10^9 ns(1.39s)$	 $1.37 \cdot 10^9 ns(1.37s)$
Résolution système			$4.66 \cdot 10^{12} ns(1h17)$
Chargement système			$3.78 \cdot 10^9 ns(3.78s)$
Mise à jour	$1.6 \cdot 10^8 ns(160ms)$	$1.52 \cdot 10^8 ns(152ms)$	$1.03 \cdot 10^9 ns(1.03s)$

FIGURE 46 – **Benchmark sur différentes résolutions de grille**
Champ de distance borné d'une capsule. Déformée par une cage de 26 sommets et 48 faces. Rendu par un plan de coupe de résolution 1024×1024 *soit* 2^{20} *évaluations.*

Quatrième partie

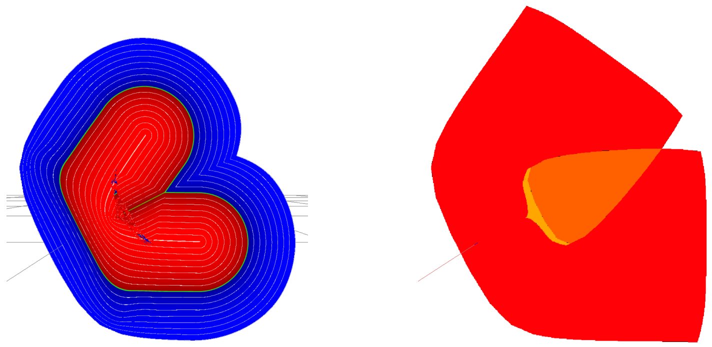
Corrections du maillage et pistes de recherche

Cette partie présente les approfondissements de mes travaux de stage. C'est-à-dire que je présenterai mon étude de notre méthode dans les cas d'auto-intersections de repliements de l'espace. L'intérêt de ces travaux est de gérer ensuite une correction du maillage lorsqu'il entre en contact avec lui-même.

11 Auto-intersections et correction du champ scalaire

11.1 Repliement de l'espace et résolution du champ scalaire

L'intérêt des surfaces implicites est de pouvoir les composer entre elles, ce qui permet notamment de proposer la gestion d'un pli par l'Implicit skinning. Ce que nous voulons proposer est de permettre cette composition mais cette fois en utilisant une seule surface implicite. Ce sont les déformations par la cage qui vont amener la surface à s'auto-intersecter ce qui rend le travail du champ scalaire intéressant lorsque ceci arrive. Nous voulons en particulier gérer les cas d'auto-contact : lorsqu'une surface se touche superficiellement, mais aussi les cas des auto-intersections : c'est-à-dire principalement les plis lorsque la cage est tordue de sorte qu'elle se plie en entre en contact.



(a) Champ déformé sans ordre sur les antécédents (b) Nombre d'antécédents détectés

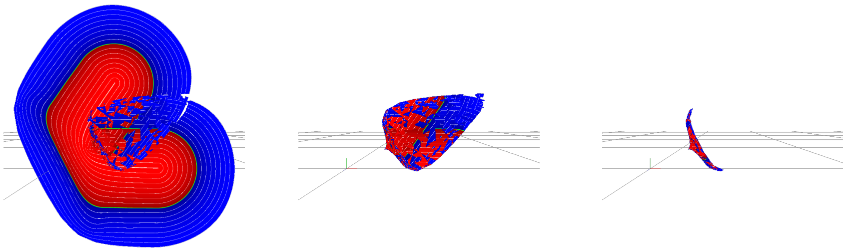
FIGURE 47 – **Problème de non associativité de l'opérateur de contact**
En (a) le champ reconstruit en résolvant les régions à antécédents multiple par l'opérateur de contact en champs de distance de l'Implicit skinning. En (b), le nombre d'antécédents détectés (en rouge : 1 antécédent, en orange : 2 antécédents et en jaune : 3 antécédents).

Nous proposons donc de corriger le champ scalaire déformé dans les cas d'antécédents multiples en utilisant l'opérateur de contact de l'Implicit skinning. La première observation a été de voir que lorsque l'on plie la cage, un phénomène d'interférence se produit ce qui n'est pas le cas lors de la composition de deux surfaces (figure 47). En effet, dans cette région, nous avons 3 antécédents. Le problème posé est l'ordre des antécédents (figure 48). En particulier que cet ordre dépend du parcours dans la hiérarchie d'intervalles englobants. Ceci fait qu'il est même amené à changer lors que la cage est déformée, par suite lorsque la hiérarchie est refabriquée.

En effet dans le cas d'un opérateur comme le max, l'ordre des éléments importe peu puisque l'opérateur est associatif. Mais l'opérateur de contact proposé pour l'Implicit skinning ne l'est pas :

$$\texttt{contact}\left(a, \texttt{contact}\left(b, c\right)\right) \neq \texttt{contact}\left(\texttt{contact}\left(a, b\right), c\right).$$

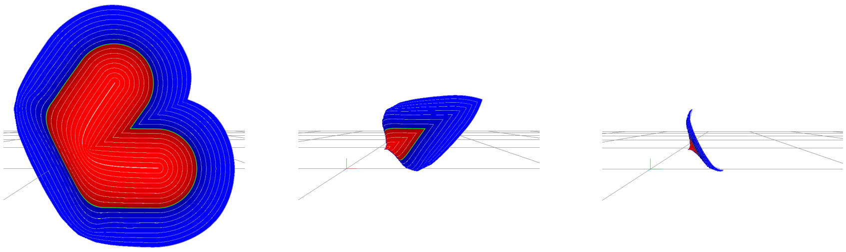
Cependant dans la zone où nous détectons 2 antécédents, puisque l'opérateur est symétrique l'ordre ne pose pas de problème et compose les valeurs scalaires comme on le ferait pour deux surfaces. Car en



(a) Premier antécédent (b) Second antécédent (c) Troisième antécédent

FIGURE 48 – **Valeurs associées aux antécédents**
Pour représenter les valeurs obtenues pour chaque antécédent, nous proposons de donner les champs scalaires obtenus pour chaque antécédent selon l'ordre dans lequel ils sont donnés.

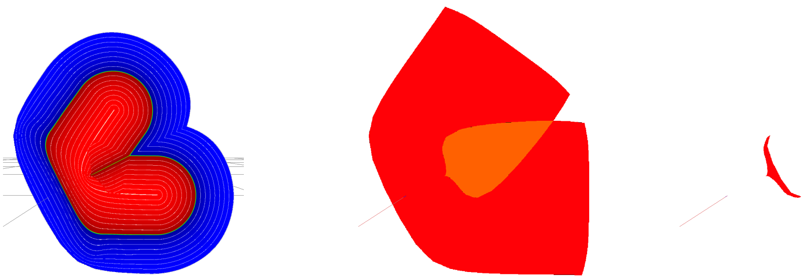
effet dans le cas de surfaces implicites, l'ordre est implicitement donné par l'ordre de composition des surfaces. Tandis qu'ici, nous n'avons pas cette propriété.



(a) Premier antécédent (b) Second antécédent (c) Troisième antécédent

FIGURE 49 – **Valeurs associées aux antécédents ordonnés implicitement**
L'ordre implicite proposé ordonne les éléments par ordre décroissant selon leur valeur scalaire dans le champ d'origine. En particulier, la première tranche d'antécédents (a) correspond à la valeur maximale.

Ceci a conduit à proposer d'imposer un ordre implicite aux antécédents (figure 49), c'est-à-dire qu'on les classe en fonction de leur valeur scalaire dans l'ordre décroissant. Ceci dans le but de composer en premier les antécédents de plus grande valeur scalaire puis composer le résultat récursivement avec les suivants (figure 50).



(a) Champ avec contact propre (b) Antécédents orientés (c) Antécédents renversés

FIGURE 50 – **Valeurs associées aux antécédents**
En (a) le résultat obtenu depuis la composition par l'ordre implicite et est équivalent à composer les deux plus grandes valeurs ou ignorer l'antécédent dans un tétraèdre renversé. En (b), la représentation du nombre d'antécédents issus de tétraèdres non renversés et en (c) ceux issus d'un tétraèdre renversé.

En réalité, dans ce cas, composer toutes les valeurs scalaires lors d'un pli ou seulement les deux plus grandes revient ici au même. En effet, la troisième valeur est négligeable à côté des deux autres ce qui fait que l'opérateur agit comme un max avec cette troisième valeur et non comme un contact. De plus, nous nous sommes aussi intéressés à regarder l'orientation d'un tétraèdre. Dans le cas où le déterminant de la matrice utilisée pour la détection est de signe différent à celui calculé pour le tétraèdre d'origine, alors le tétraèdre a changé d'orientation. Nous le considérons donc comme renversé. Il est alors intéressant de voir, qu'en réalité le troisième antécédent vient d'une partie où les tétraèdres ont été renversés (leur détection est illustrée en figure 50). Je nommerai cette partie l'espace renversé $\mathfrak{R}$ et la partie de l'espace qui conserve l'orientation des tétraèdres l'espace orienté $\mathfrak{O}$. De plus, si on compose uniquement les antécédents de l'espace renversé on obtient le même résultat que pour l'ordre implicite.

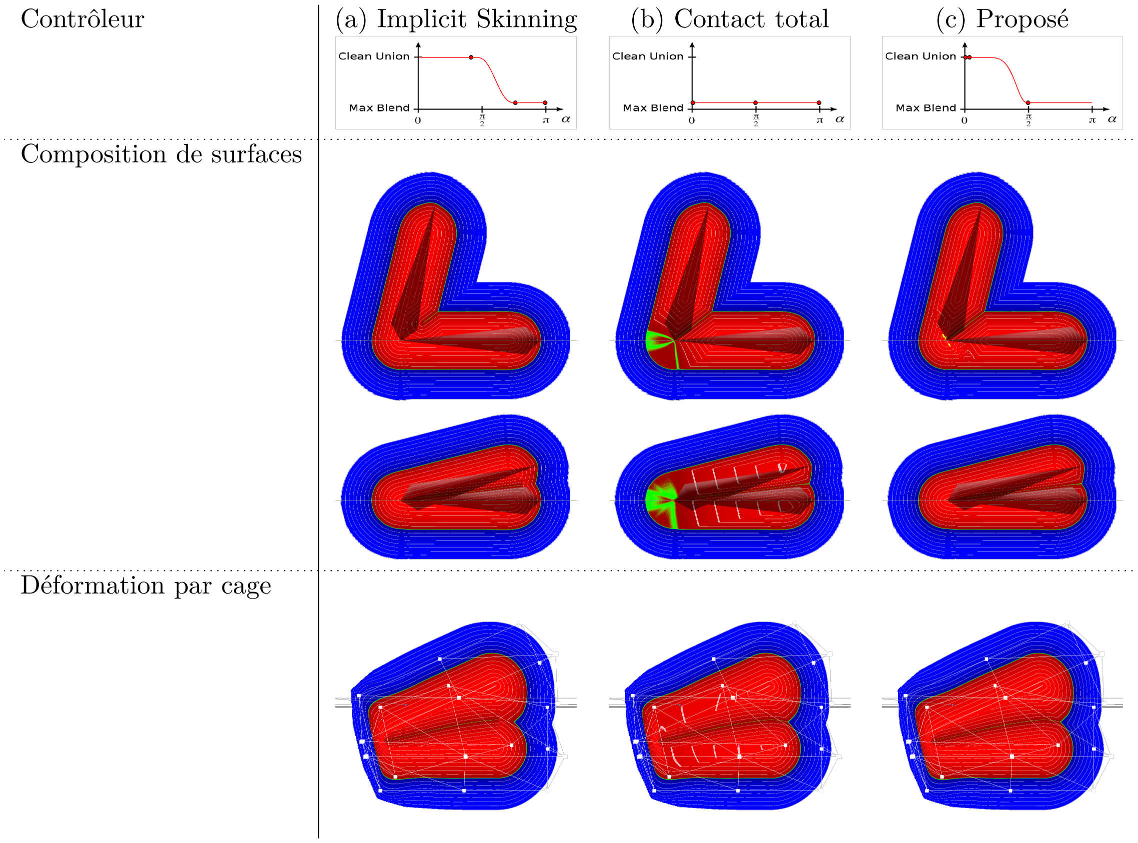
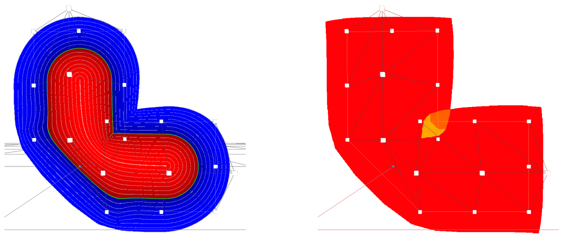


FIGURE 51 – **Contrôleur proposé pour l'opérateur de contact**
Le changement du contrôleur de l'opérateur basé gradient où Clean Union correspond à ce que l'opérateur prenne l'aspect d'union et Max Blend correspond à l'opérateur de contact en champs de distance. En premier est donné l'aspect du contrôleur, puis l'aspect de la composition que ceci donnerait sur deux surfaces pour l'Implicit skinning basé squelette et finalement le résultat pour la déformation par cages.

L'opérateur basé gradient de l'Implicit skinning doit faire un compromis entre permettre le contact pour créer un pli sans qu'il soit trop important lorsqu'il s'amorce et conserver le contact lorsque les surfaces s'écrasent l'une contre l'autre (figure 51). Ceci fait que par défaut, pour un contact trop important dans les zones arrondies, le contact devient une union classique. L'un des avantages que nous avons ici est que la composition par un opérateur est effectuée uniquement lorsqu'il y a détection multiple. Ceci permet par exemple de pousser le contact plus loin tout en restant protégé par la détection où l'amorçage du pli ne change pas (figure 52).

Nous avons donc essayé un opérateur de contact non basé gradient mais le problème posé est qu'il fait disparaître le squelette topologique et est effectué des deux côtés de celui-ci (figure 51). Ce qui



(a) Champ préservé (b) Détection d'antécédents

FIGURE 52 – **Préservation du champ par la détection d'antécédents multiples**

est normal par la définition de l'opérateur et le fait qu'il ne soit pas basé gradient (puisque c'est l'angle entre les gradients qui permet d'éviter que la partie de l'autre côté du squelette topologique soit affectée).

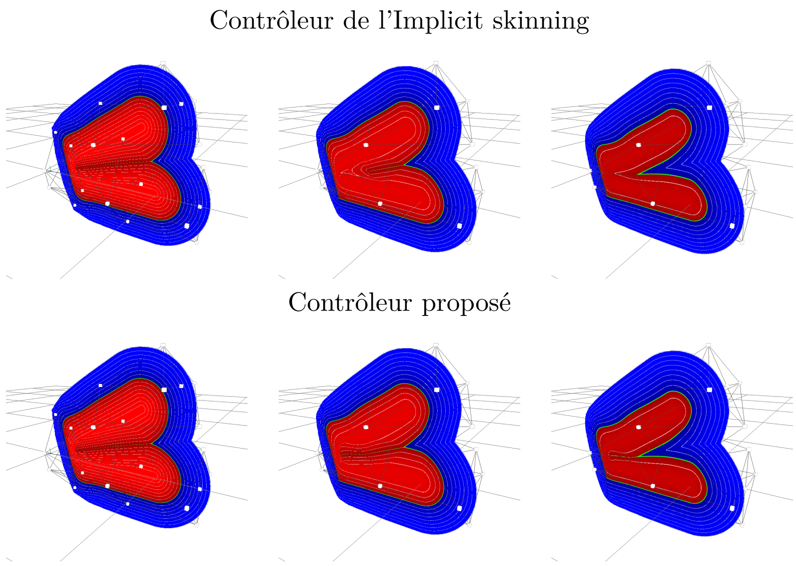


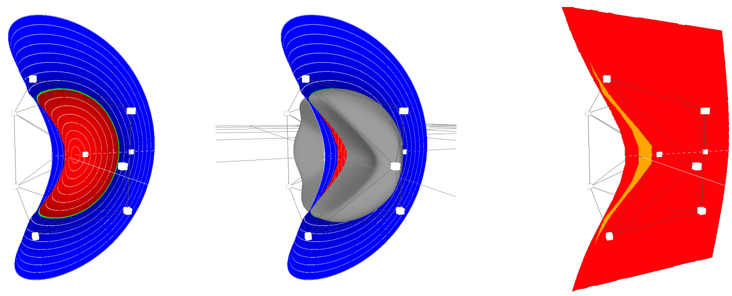
FIGURE 53 – **Conservation du contact en 3D**

Une modification de l'opérateur de contact proposée dans le cas de la déformation par cages consiste donc à régler le contrôleur de l'opérateur basé gradient de sorte de proposer une meilleur conservation du contact (figure 51). De plus cette modification est intéressante car elle permet une meilleure conservation du contact en 3D comme illustré en figure 53.

11.2 Contraction de l'espace et résolution

Certaines contractions de l'espace exagérées peuvent l'amener à s'intersecter avec lui-même. Dans le cas de la déformation d'un maillage ceci ne pose pas de problème puisqu'il suit la déformation et crée un concavité. Mais dans le cas de notre champ, par défaut son iso-surface en utilisant l'opérateur max ne suit pas la déformation du maillage par le système de coordonnées (figure 54). Ceci est dû à la formation d'une zone à antécédents multiples.

La solution que nous proposons se base à la fois sur l'idée de soustraire un champ scalaire à un autre ce qui se fait par l'opérateur de différence. Pour se le représenter : l'opérateur max fait l'union de deux champs. Par suite, notre champ est dans $[0, 1]$ et l'intersection correspond au complémentaire de l'union



(a) Champ déformé (b) Correspondance (c) Nombre d'antécédents

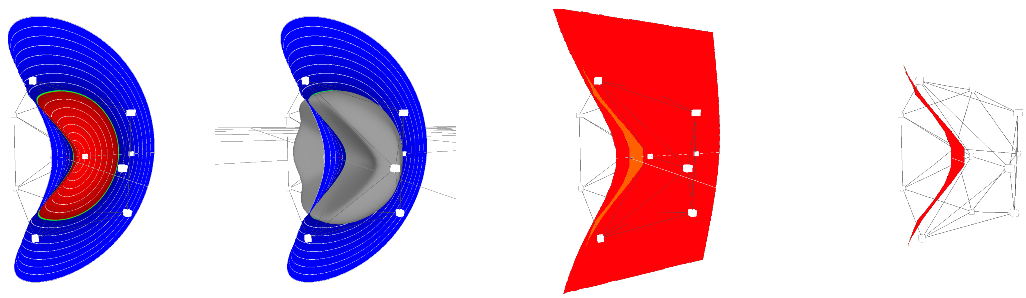
FIGURE 54 – **Champ scalaire contracté résolu avec l'opérateur max**

des complémentaires, ce qui se traduit par l'opérateur suivant :

$$1 - \max(1 - a, 1 - b) = \min(a, b).$$

Par suite, la différence est l'intersection avec le complémentaire, ce qui nous donne :

$$\texttt{diff}(a, b) = \min(a, 1 - b).$$



(a) Champ déformé (b) Correspondance (c) Antécédents orientés (d) Antécédents renversés

FIGURE 55 – **Champ scalaire contracté résolu avec classification et opérateur de différence**

Notre configuration se divise en 3 parties :
- L'espace orienté peu affecté par la contraction d'un côté de la contraction.
- L'espace renversé qui suit la déformation du sommet et donc qui se renverse du fait que la déformation soit trop large et fait se renverser l'espace pour s'auto-intersecter.
- L'espace orienté moins affecté par la contraction de l'autre côté de celle-ci.

La proposition ici est d'admettre que les antécédents d'un tétraèdre orienté forment un seul espace et donc de les composer par l'opérateur max tel que proposé initialement. Mais que la partie de l'espace renversé est provoqué par la contraction de l'espace sur lui-même et donc de soustraire cette partie à l'espace orienté d'où utilisation de l'opérateur de différence :

$$g(Q) = \texttt{diff}(\max\{f(P)\,|P \in \mathfrak{O}\}, \max\{f(P)\,|P \in \mathfrak{R}\}).$$

Cette proposition permet d'obtenir un champ plus cohérent dans le cas de contractions de l'espace (figure 55). Cependant, le champ proposé aux contractions peut s'avérer problématique pour la composition : en effet, ici le champ à l'extérieur de la surface est écrasé près de l'iso-surface ce qui le rend problématique pour la composition. Une projection ne poserait pas problème en entrant dans cette zone à partir du moment où elle prend en compte la norme du gradient, mais une utilisation de ce gradient pour une détermination du plan tangent peut ici poser problème. Cependant une observation intéressante pour la suite est de remarquer qu'il existe deux lignes d'interférences dans le champ qui existent aux frontières entre les espaces orienté et renversé.

12 Implicit skinning et déformation par les cages

12.1 Adaptation de la déformation pour l'Implicit skinning

Pour effectuer l'étape de projection, ceci demande de connaître le gradient du champ scalaire pour un antécédent. Le gradient de la surface déformée g est donné depuis la transformation du système de coordonnées T et le gradient de la surface d'origine f par l'expression suivante :

$$\nabla g\left(Q\right) = J_T\left(P\right) \cdot \nabla f\left(P\right)$$

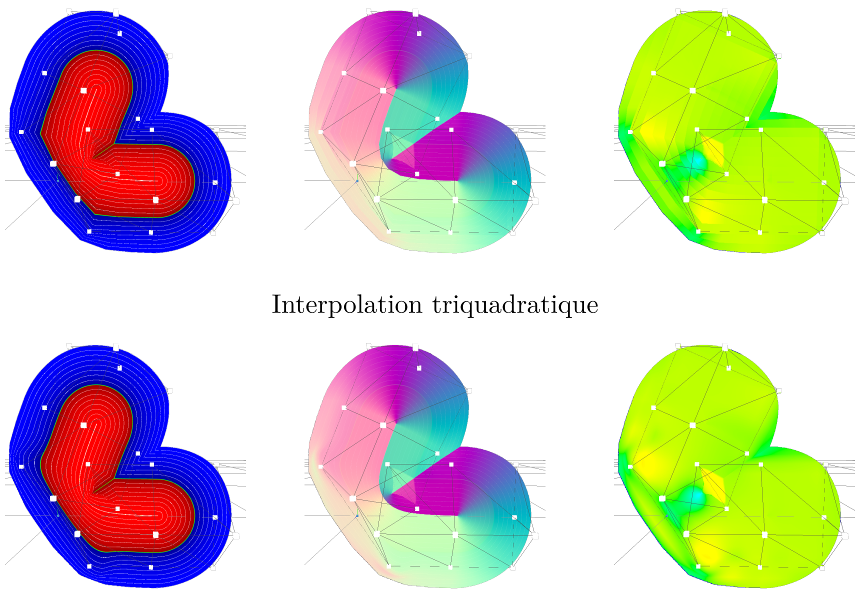
où Q est la position évaluée du champ déformé et P son antécédent. Pour simplifier l'explication nous supposons un antécédent, mais les gradients se composent depuis les dérivées de l'opérateur dans les cas d'antécédents multiples. Le problème posé ici est que calculer J_T suppose d'utiliser le *binding* du système de coordonnées. Nous proposons donc de reconstruire le gradient depuis l'expression de la jacobienne d'approximation suivante :

$$J_T\left(P\right) = J_R\left(\mu\left(P\right)\right) J_\mu\left(P\right)$$

où R est la transformation utilisée pour la reconstruction (trilinéaire, triquadratique ou autre) et μ est la fonction qui converti une positon P de l'espace d'origine en les coordonnées paramétriques utilisées pour la reconstruction R. Nous illustrons en figure 56 la reconstruction du gradient par l'interpolation trilinéaire et triquadratique. À noter qu'il est possible de remarque des légères discontinuités de la norme et de la direction du gradient dans le cas de l'interpolation trilinéaire ce qui est corrigé par l'interpolation triquadratique.

Interpolation trilinéaire

Interpolation triquadratique



(a) Champ scalaire (b) Direction du gradient (c) Norme du gradient

FIGURE 56 – **Reconstruction du gradient**
Nous comparons les champs (a) et gradients (b)(c) obtenus pour les reconstructions linéaire et quadratique. Pour (b), la couleur représentée donne une information sur la direction du gradient. En (c), on représente sa norme : en bleu les normes de l'ordre de $\frac{1}{4}$ ou moins, en rouge les normes de l'ordre de 4 ou plus, le vert clair correspond à une norme 1.

De plus, la matrice jacobienne correspond à la matrice de rotation et changement d'échelle. Je propose donc de remplacer dans l'Implicit skinning la matrice de rotation obtenue depuis le squelette d'animation pour l'ARAP par cette matrice jacobienne.

Nous basons notre correction sur l'Implicit skinning [Vaillant et al., 2014] où la déformation des sommets est donnée par le déplacement relatif de chaque articulation. Ici, la déformation par cage se base sur la position dans la cage d'origine pour donner une position d'un sommet à tout instant. Cependant dans le cas d'un sommet corrigé, utiliser la déformation par les cages telle qu'elle reviendrait à l'Implicit skinning proposé initialement [Vaillant et al., 2013].

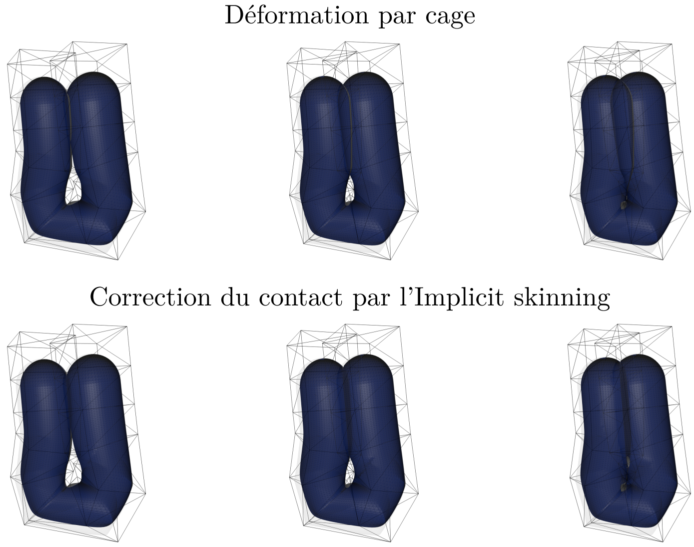


FIGURE 57 – **Correction automatique d'un auto-contact du maillage**

Nous proposons donc de considérer la transformation à un temps t : $T^t(P) = \sum_{i\in[n]} \alpha_i(P) D_i^t$ la correspondance d'un sommet P dans l'espace interpolé par la cage $\{C_i\}_{i\in[n]}$ vers l'espace interpolé par une cage $\{D_i^t\}_{i\in[n]}$. En connaissant la cage à la frame précédente $\{D_i^t\}_{i\in[n]}$ et la cage à la frame actuelle $\{D_i^{t+1}\}_{i\in[n]}$ nous proposons de déplacer un sommet corrigé à la frame précédente Q_i^t en lui appliquant la somme pondérée par les coordonnées barycentriques généralisées des translations de $\{D_i^t\}_{i\in[n]}$ vers $\{D_i^{t+1}\}_{i\in[n]}$, ce qui nous donne :

$$Q_i^{t+1} = Q_i^t + \sum_{i\in[n]} \alpha_i(P_i)\left(D_i^{t+1} - D_i^t\right)$$

où P_i est la position initiale bindée dans la cage d'origine. De plus, en réalité ce calcul s'écrit plus simplement :

$$Q_i^{t+1} = Q_i^t + T^{t+1}(P_i) - T^t(P_i).$$

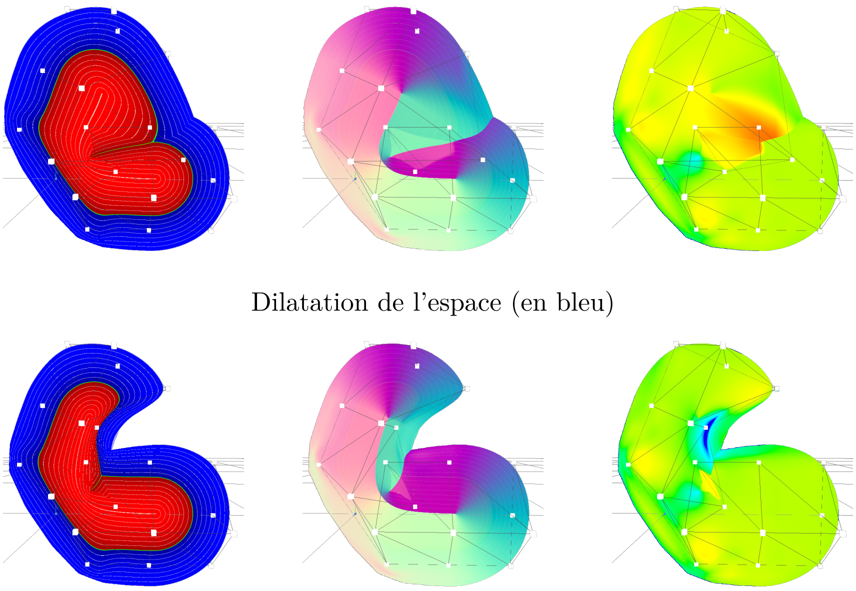
C'est ensuite cette position Q_i^{t+1} qui sera considérée comme la position du sommet à corriger si le sommet reste dans la région à antécédents multiples. Dans le cas où une position n'est pas dans la région à antécédents multiples, nous considérons simplement le sommet déformé par la cage $T^{t+1}(P_i)$ et aucune correction n'est effectuée. Ceci permet de proposer la correction aux contact comme illustré en figure 57.

12.2 Changement d'échelle locale du champ scalaire

La reconstruction du gradient proposée permet cependant de récupérer un gradient dont la norme dépend des contractions et dilations de l'espace comme illustré en figure 58.

Ceci est intéressant, car le gradient non normalisé permet une projection qui reste pertinente. Cependant cette information de norme reste locale et n'est pas toujours exacte dans le cas d'une déformation trop large ce qui fait que la projection peut ne pas atteindre sa cible directement. En effet, il faudrait cette information de changement de norme à la position qui va être projetée, mais bien souvent la déformation est effectuée sur la surface et donc souvent locale à celle-ci. Ceci fait que pour une position projetée

Contraction de l'espace (en orange)



(a) Champ scalaire (b) Direction du gradient (c) Norme du gradient

FIGURE 58 – **Reconstruction du gradient**

éloignée de la surface, la norme du gradient est moins impactée par la déformation et donc la norme du gradient moins impactée. En général, c'est peu visible car le maillage reste proche de l'iso-surface même pour un champ corrigé. Mais une déformation large qui passerait d'une zone non corrigée à une zone de contact important peut ne pas proposer la correction attendue. Mais nous supposons ici que l'animateur recherche des déformations fluides et non saccadées. Par exemple, ceci permet de proposer de gérer à la fois un pli et un effet de muscle en utilisant la cage (figure 59).

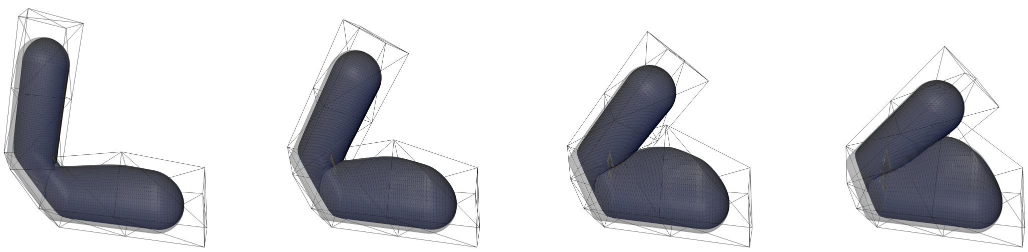
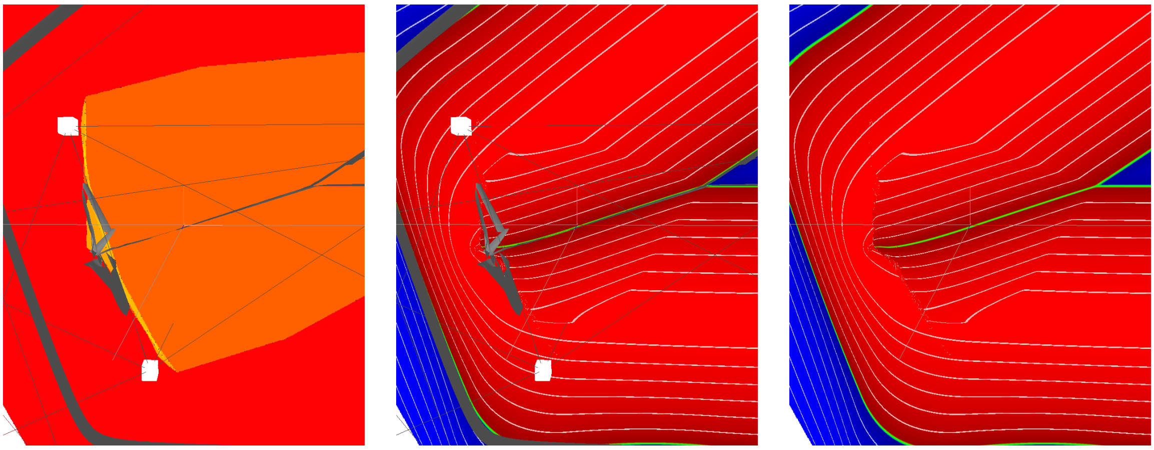


FIGURE 59 – **Effet de muscle créé et exagéré par la cage lors d'un pli**

12.3 Auto-intersection : problème du repliement de l'espace

En section 11.1 nous avons observé un phénomène étrange où dans le cas d'un pli nous avons 3 antécédents dont 1 antécédent qui appartient à l'espace renversé. Ici, ceci est un problème pour proposer une correction propre par Implicit skinning pour un pli (figure 62). En effet, nous observons qu'une partie du maillage touche la région qui sépare la partie où 1 antécédent est détecté et celle où 3 antécédents sont détectés.

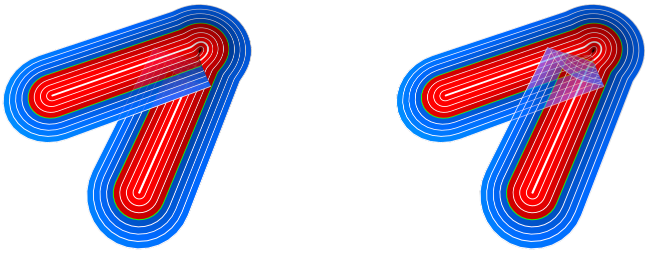
Sur la figure 62 nous pouvons observer qu'une partie du maillage corrigé est dans la zone à 1 antécédent. De plus, le maillage corrigé à la frontière entre la zone où existent plus de 3 antécédents et 1 antécédent est projeté de manière non pertinente ce qui peut même aller hors de la surface d'où la



(a) Détection multiple (b) Champ et problème (c) Champ scalaire

FIGURE 60 – **Problème de correction observé au pli**
En (a), le maillage corrigé et le nombre d'antécédents. En (b) la superposition du maillage corrigé avec le champ scalaire et en (c) le champ scalaire déformé.

correction en imposant la déformation par cage classique lorsque cela arrive. Ce problème est dû aux interférences dans le champ scalaire à la frontière entre ces deux régions. En effet, le gradient est fortement discontinu en cet endroit ce qui pose des problèmes de projection des sommets.

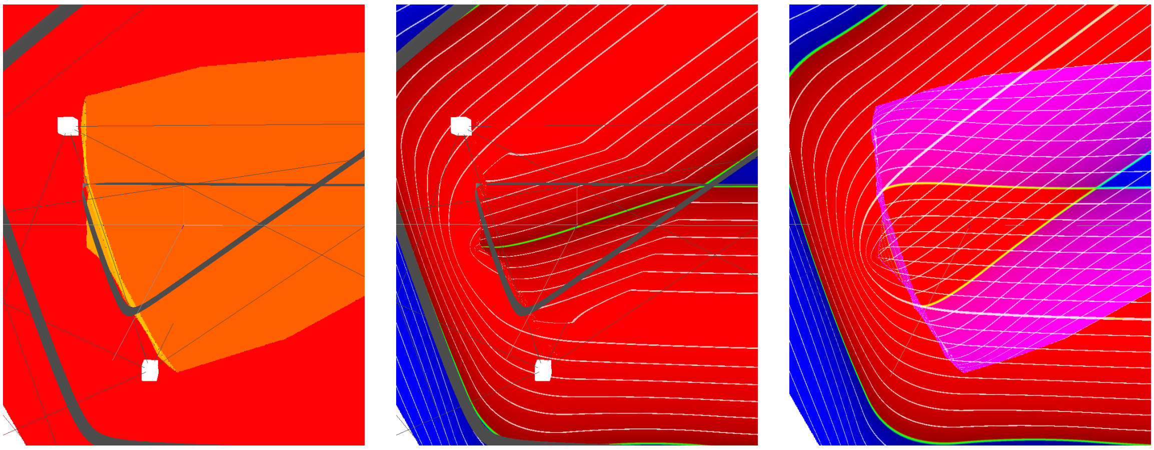


(a) Déformation de la capsule (b) avec transparence

FIGURE 61 – **Concept du repliement de l'espace**
En (a) la déformation d'une tranche de la capsule comme une surface pliable. En (b) la visualisation en transparence des différentes parties de la surface (une orientée, une renversée puis une orientée). Ceci correspond à la superposition des tranches des valeurs des antécédents sur une tranche de l'espace.

Pour essayer d'expliquer ce qu'il se passe, ma vision de la configuration est d'imaginer que sur une tranche, le repliement de l'espace sur lui-même ce comporte comme le ferait un tissu comme je l'illustre en figure 61. C'est-à-dire que pour plier un tissu, ceci demande de superposer sur une face, une face renversée de celui-ci puis au dessus une autre face dans la même orientation. C'est le même type de phénomène qui se produit dans le cas de la contraction de l'espace de la section 11.2. Ce qui fait la ligne d'interférence est la frontière entre une partie de l'espace orienté et une partie de l'espace renversé. En effet si on imagine écraser la représentation en figure 61 sur une seule tranche, le gradient serait potentiellement discontinu sur cette frontière.

De plus, en réalité nous n'avons pas tout le champ scalaire pour reconstruire la surface. En effet, nous nous basons sur l'idée que notre grille discrète capture la déformation du système de coordonnées. Or, ceci ne permet pas nécessairement de capture la frontière entre les espaces déformés et renversés. En réalité, il me semble difficile voir non possible de définir un échantillonnage discret qui soit en mesure de capturer toutes les frontières que l'on peut fabriquer puisqu'elles existent par la déformation crée et n'ont aucune raison de suivre un échantillonnage donné. Notre grille de détection est donnée par des arêtes droites ce qui ne peut pas permettre la capture potentiellement courbée de notre frontière.



(a) Détection multiple (b) Champ et maillage (c) Superposition des champs

Figure 62 – **Problème de création d'une auto-intersection par pli**
On propose la visualisation du maillage d'origine avec le décompte des antécédents (a) et le champ scalaire utilisé (b). Pour visualiser la déformation de la surface implicite et les outils donnés pour corriger le champ, nous superposons en (c) l'ensemble des champs scalaires pour les différents antécédents obtenus.

De plus, pour créer le phénomène d'auto-intersection lors d'un pli, ceci demande à ce que la surface forme une boucle comme on peut l'observer et la correction repose sur l'écrasement de celle-ci. Or, pour créer cette boucle, ceci demande à ce que la surface se renverse et passe par l'espace renversé. Par conséquent, la surface à corrigée passera nécessairement par la frontière entre un espace orienté et renversé. Ceci met en échec notre correction par l'Implicit skinning même dans le cas incrémental.

12.4 Benchmark et limitations

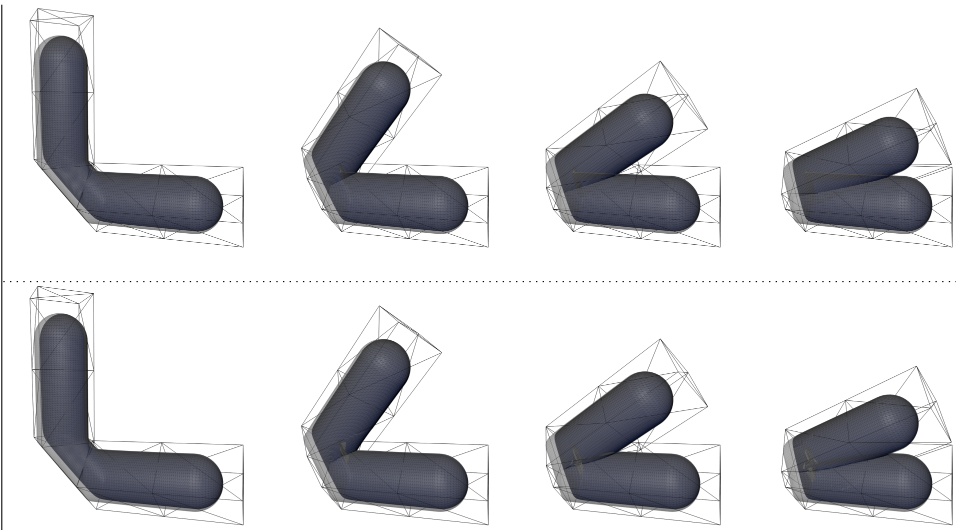
Pour donner une idée des performances actuelles de notre méthode, nous comparons ses résultats sur un pli avec la déformation par cage sans correction. Pour une déformation par cage sur cette même capsule, ceci demande de l'ordre de $2.8 \cdot 10^6 ns (2.8ms)$. La grille utilisée est en résolution $16 \times 16 \times 16$ et la reconstruction utilisée est l'interpolation triquadratique ce qui donne un temps de mise-à-jour de l'ordre de $6.8 \cdot 10^7 ns (68ms)$. Nous proposons en figure 63 un comparatif de notre méthode contre la déformation par cage classique. Le facteur de temps considéré est le ratio entre le temps demandé par notre correction et la déformation par cages classique.

De plus, pour le moment notre méthode souffre principalement des défauts suivants :

— Des erreurs de correspondance entre la déformation du maillage et la déformation de la surface implicite. En effet la création de grossissements aux régions coudées par l'interpolation triquadratique peut créer des sauts sur le maillage lors de déformations trop larges qui font qu'une région à antécédents multiples intersecte cette partie du champ. De plus, le compromis entre la taille des cellules et le temps de mise-à-jour fait que bien souvent l'échantillonnage est trop grossier et les cellules trop larges, ce qui ne permet pas de capturer les plis que peut faire le maillage et par conséquent les projeter sur l'iso-surface ce qui ne serait pas souhaité ici (figure 64).

— Le problème d'interférences dans le champ dû aux frontières entre un espace orienté et un espace renversé détaillé en section 12.3.

— Un problème de cohérence du maillage sur l'iso-surface. En effet, la résolution de l'ARAP uniquement dans les régions à détection d'antécédents multiples semble provoquer un problème de cohérence à la frontière entre une zone corrigée et un zone évaluée par la déformation par cages (figure 64). À noter que la résolution de l'ARAP demande l'utilisation du champ scalaire pour récupérer le gradient et l'utiliser pour déterminer un plan tangent. Ceci fait que pour utiliser un ARAP local, il faudrait travailler sur l'optimisation du temps demandé pour l'évaluation du gradient du champ déformé. Bien que ce problème soit encore à étudier.

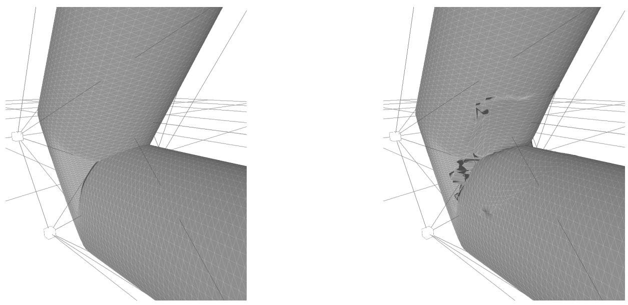


Par cage :

Implicit skinning :

Sommets corrigés :	0	1117	4481	7263
Temps :	$6.9 \cdot 10^{7} ns$ ($69ms$)	$2.35 \cdot 10^{8} ns$ ($235ms$)	$4.84 \cdot 10^{8} ns$ ($484ms$)	$5.58 \cdot 10^{8} ns$ ($558ms$)
Facteur temps :	24	83	172	200

FIGURE 63 – **Benchmark de la correction d'un pli par l'Implicit skinning et notre champ déformé**
Les résultats des déformations par une cage et par notre méthode avec l'Implicit skinning. En transparence la partie visible du maillage ce qui laisse apparaître la l'accentuation du pli par notre méthode ainsi que l'écrasement du maillage à l'intérieur pour former le contact. Un benchmark est aussi proposé en indiquant le nombre de sommets corrigés, le temps demandé par notre méthode et le facteur de temps supplémentaire demandé comparé à la déformation par cage des sommets.



(a) Déformation par cage (b) Corrigé par l'Implicit skinning

FIGURE 64 – **Problèmes de correction**
Résultat d'un dépliement de la capsule où le maillage perd sa cohérence sur la frontière entre la région corrigé et non corrigée. Effet de projection des sommets d'un pli formé par le système de coordonnées sur l'iso-surface de la surface implicite (pli non représenté dans le champ scalaire à cause d'un manque de précision de l'échantillonnage).

13 Pistes de recherche et idées de travaux futurs

13.1 Opérateurs de grossissement et d'étirement pour effets physiques automatiques

Offrir la possibilité de gérer des petits effets physiques automatiquement aux zones de contact pourrait rendre notre méthode de déformation par cage particulièrement intéressante en plus de corriger les plis comme le fait l'Implicit skinning. Ceci pourrait permettre à un animateur de gérer des déformations contrôlées par la cages mais l'intérêt pour notre méthode se baserait sur l'ajout de grossissements pour simuler le contact avec du tissu adipeux ou d'étirements automatiques du maillage pour simuler l'écrasement d'un muscle ou organe à l'aide d'opérateurs définis sur le surfaces implicites. En figure 65 nous proposons un exemple ou on pourrait vouloir exagérer un effet de muscle et exagérer le pli avec grossissement sur le bras d'un personnage enrobé.

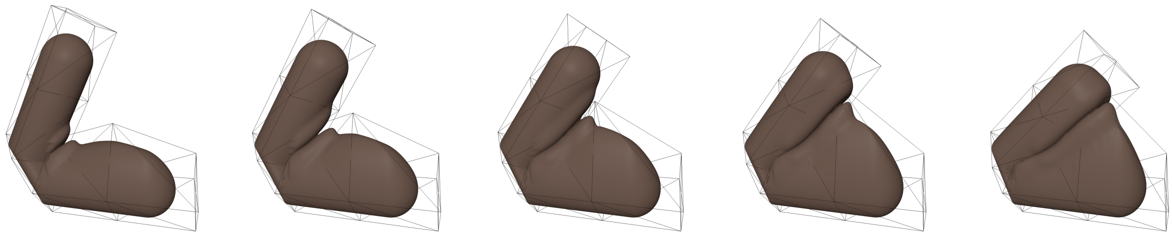


FIGURE 65 – **Effet de muscle exagéré par la cage et correction automatique par un pli avec grossissement par le champ scalaire**

Ceci demande en particulier de travailler sur le champ scalaire (figure 66). En particulier, pour proposer cette fonctionnalité, ceci nous demanderait dans un premier temps de travailler sur la définition d'opérateurs de grossissement et de tension basés champs de distance. Puis de trouver un moyen pour laisser un infographiste paramétrer l'effet qu'il souhaite obtenir (potentiellement en le laissant peindre la surface). La difficulté posée ici est de trouver une manière de calculer notre opérateur efficacement en fonction des effets demandés. À noter qu'un opérateur basé gradient est pré-calculé dans une grille $128 \times 128 \times 128$ ce qui correspond en mémoire à l'avoir en RAM pour un coût d'environ 10 Mo. Ajouter une dimension et utiliser la même méthode demanderait de l'ordre de 1 Go en RAM.

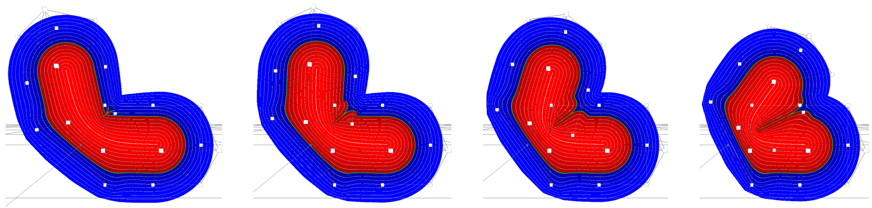


FIGURE 66 – **Champ scalaire utilisant un opérateur de grossissement**

13.2 Proposition de troisième approche : reconstruction d'un champ de distance

Bien que la seconde méthode proposée soit utilisable pour l'Implicit skinning comme illustré précédemment. Elle présente des défauts de l'ordre du temps de calcul pour la mise-à-jour ce qui limite la résolution de la grille, par suite elle demande de prendre une grille trop grossière ce qui provoques des effets de non concordance avec la déformation du système de coordonnées comme des grossissements. De même, cette méthode ne permet pas la correction totale du pli pour cause de problèmes intrasèques à l'interpolation de l'espace mais aussi des parties non capturées de l'iso-surface (aux frontières entre espace orienté-renversé). De plus à l'heure actuelle, nous sommes dépendant du fait de demander au système de coordonnées d'interpoler l'extérieur de la cage ce qui limite les systèmes de coordonnées utilisables. Bien qu'on puisse améliorer cette méthode, je pense qu'il est pertinent de s'intéresser à une nouvelle méthode. Je propose en annexe D un nouveau concept qui se base sur l'idée de capturer la déformation de la surface et reconstruire un champ de distance.

Cinquième partie
Bilan personnel et professionnel

L'objectif de mon stage était de proposer la déformation d'une primitive implicite pour être en mesure d'ajouter des effets tels que des gonflements à l'Implicit skinning qui est initialement basé sur des transformations rigides. Pour répondre à cette problématique, j'ai proposé un architecture en 63 modules répartis en environ 12 000 lignes de code `C++` dans une branche dédiée sur le `gitlab` privé de l'équipe STORM. J'ai de même produit un prototype de l'ordre de 3 800 lignes en `Java` sous *Processing IDE* dans un premier temps pour m'initier à la résolution de la position inverse des *Free Form Deformations* puis l'implémentation de la méthode de résolution numérique avant de l'implémenter en 3D sous *Radium Engine*. Les résultats de mes travaux, bien que non temps réel, proposent la déformation de primitives implicites par une cage de déformation.

De plus, mes travaux ont mené à s'intéresser la composition d'une surface implicite avec elle-même. Ce qui a mené à ce que mes travaux soient aussi suivi par la suite par Florian Canezin et Nicolas Mellado (chercheurs à l'IRIT), puis ensuite par Brian Wyvill (Université de Victoria : Victoria, Canada) et Paul Kry (Université de McGill : Montréal, Canada). Par la suite l'idée est d'utiliser cette surface implicite pour proposer une correction propre du maillage. Ceci peut mener à la publication d'un article de recherche lorsque mes travaux seront en mesure d'atteindre cet objectif.

En effet, en plus de compétences scientifiques, l'équipe STORM m'a aussi donné l'occasion de présenter mes travaux en anglais lors d'un séminaire en Juin. Puis, par la suite, Loïc a ouvert un un groupe de discussion sur Slack pour me permettre de présenter régulièrement les avancées de mes travaux ce qui a aussi été l'occasion d'avoir des échanges enrichissants et stimulants pour mes travaux. De même, mon sujet de stage ouvrant la possibilité d'explorer plein de pistes différentes, c'était aussi l'occasion de m'aider à choisir quelle piste privilégier. Pour donner un exemple : j'ai privilégié la piste de la reconstruction basée sur les tétraèdres après une visio avec Paul Kry et Loïc Barthe ce qui a conduit à une reconstruction du champs plus propre.

Pour mener à bien mon stage, ma formation en informatique m'a donné les outils me familiariser plus facilement avec un projet tel que *Radium Engine*. De plus j'ai pu m'appuyer sur des bases données par le cours de géométrie différentielle discrète et le cours d'optimisation de ma formation à l'Institut Gaspard Monge, ainsi que des connaissances sur la géométrie, le calcul matriciel et la résolution numérique de ma licence en mathématiques mais aussi mes connaissances personnelles sur l'infographie en particulier sur la modélisation 3D et l'animation de maillages.

De plus, j'ai pu aborder des thématiques qui complètent ma formation en sciences de l'image. En effet, ce stage m'a donné l'occasion de découvrir la théorie des surfaces implicites qui semblait une théorie intéressante et amusant au début de mon stage jusqu'à m'y familiariser par la pratique : aussi bien la composition de surfaces implicites par un opérateur basé gradient que le paramétrage de celui-ci pour résoudre les auto-intersections par des contacts. De plus, j'ai aussi eu l'opportunité de contribuer à une méthode basée cage de l'Implicit skinning ce qui m'a permis d'étudier et comprendre plus en détail le fonctionnement de l'Implicit skinning pour proposer une adaptation des concepts pour les cages.

Au cours de ma formation à l'Institut Gaspard Monge j'ai eu l'occasion de travailler sur des projets où le but est de produire un livrable sur la base sur de fonctionnalités attendues. Là où mon stage de recherche est complémentaire car il me demande de créer un concept nouveau. Ceci laisse place à plus de créativité et d'originalité pour atteindre nos objectifs. Nous partions l'idée qui est celle de proposer la déformation d'une surface implicite par une cage de déformation. Mais pour partir dans cette direction ceci demande dans un premier temps d'acquérir les prérequis sur le sujet tels que les notions de déformation par les cages, de surface implicite et comprendre l'intérêt des travaux à rechercher pour le but final qui est de l'utiliser avec l'Implicit skinning. Puis ensuite s'intéresser à l'état de l'art sur les méthodes existantes et voir comment les utiliser et les adapter pour répondre à notre problématique.

De plus, je pense que les articles de recherche sont un bon moyen de continuer de se former après mon master. En particulier que ceci nous donne les outils les plus avancés pour continuer d'approfondir une thématique ainsi les briques de base depuis lesquelles faire avancer la recherche.

Si je devais recommencer, je pense que je suivrai une démarche scientifique similaire. En effet, je pense que dans un premier temps il faut acquérir suffisamment de connaissances et de prérequis pour avoir les outils qui nous permettrons d'avancer plus efficacement par la suite et avoir une vision assez large du sujet pour l'étudier. Cependant je pense ici que je pouvais profiter un peu plus de l'expérience et l'expertise des chercheurs qui m'entourent pour acquérir un meilleur savoir et progresser plus efficacement. Je pense que c'est ensuite par l'expérimentation qu'on avance sur le sujet une fois les bases établies. En effet, ceci permet de vérifier qu'une idée ou un concept se comporte comme attendu, mais aussi de trouver de nouveaux défis à relever pour améliorer la méthode. Par la suite, je pense qu'il est pertinent d'analyser les résultats observés pour se donner une idée de la piste la plus pertinente à suivre pour continuer d'améliorer notre méthode. C'est ce qui a été fait sur la deuxième méthode proposée. Mais aussi d'étudier les défauts de la méthode pour connaître les faiblesses qui restent intrinsèques au concept proposé. Comme le problème de temps lié au fait de passer par l'étape de *binding* dans la résolution numérique utilisée dans la première méthode et potentiellement notre détection de la frontière aux repliements de l'espace pour la seconde méthode.

Malgré tout, bien que ces méthodes ne donnent pas directement le résultat espéré, elles ont un intérêt pédagogique puisque en identifiant leurs points forts et leurs faiblesses ceci permet d'avancer proposer des méthodes de plus en plus pertinentes. C'est ce que j'espère proposer avec la troisième méthode. Cependant j'ai pu remarquer que j'avance bien plus loin et vers de directions bien plus pertinentes lorsque j'ai l'occasion d'échanger avec d'autres chercheurs. Je pense donc que profiter des idées proposées et remarques de chacun est enrichissant et permet d'explorer ou envisager des pistes que nous n'aurions pas soupçonné. En effet, pour exemple l'idée de s'intéresser aux antécédents multiples a été initié par une discussion avec Florian Canezin et peut aujourd'hui valoir le coup de partager nos travaux dans un article.

Sixième partie
Conclusion

14 Synthèse de mes travaux

Dans ce document, j'ai dans un premier temps fait état de ma compréhension de l'Implicit skinning, où j'ai posé les prérequis sur les surfaces implicites et les déformations basées squelettes. Puis, j'en suis venu à étudier les déformations basées cages et comment les utiliser pour déformer une surface implicite.

Dans un premier temps, puisqu'il n'existe pas d'état de l'art sur les déformations de surfaces implicites lorsque j'ai commencé mon stage, nous nous sommes intéressé à la déformation d'une surface implicite par les *Free Form Deformations* qui sont un type particulier de cage définie depuis un parallélépipède. Dans le même esprit, nous nous sommes intéressé à la résolution de la position inverse pour mettre en relation l'espace interpolé par la cage déformée et son système de coordonnées barycentriques généralisé avec l'interpolation de l'espace d'origine où est *bindé* la cage.

Nous proposons de considérer une surface implicite déformée comme basée sur une résolution numérique de la position inverse. Ceci s'appuie sur une architecture modulaire qui permet de brancher un système de coordonnées barycentriques généralisées choisi, pour le moment sous les hypothèses d'être aussi interpolant hors de la cage. Mais aussi de permettre un d'expérimenter plus facilement des changements de composants tels que l'échantillonnage de l'espace utilisé et la méthode de résolution numérique.

La première méthode proposée se base sur la résolution numérique directe du problème de la position inverse. Nous avons essayé deux approches à ce sujet. Une première s'intéresse à la recherche de correspondance entre l'espace cartésien où vit la cage d'origine et l'espace cartésien dont l'interpolation est donnée par la déformation de notre cage. Le problème majeur de cette approche est le temps demandé pour effectuer le résolution numérique. En effet, ceci demande d'effectuer l'étape de *binding* depuis le calcul des poids barycentriques induits par le système de coordonnées utilisé dans l'espace d'origine pour obtenir une correspondance dans l'espace déformé.

Une seconde approche proposée se basait sur l'idée de naviguer dans l'espace des coordonnées barycentriques pour obtenir la correspondance entre les deux espaces pour ne plus payer le temps de *binding* et profiter d'un temps d'évaluation de la position plus intéressant. Le problème posé ici était le manque de cohérence dans la déformation. En effet, nous pouvons observer des sauts dans le champs scalaire causés par le fait de naviguer dans un espace de coordonnées barycentriques qui n'était plus celui induit par le système choisi. D'où un champs qui était $\mathcal{C}^1$ localement à l'échantillonnage choisi.

Ces problèmes posés par la résolution numérique directe a laissé place à une méthode basée sur l'approximation de l'interpolation de l'espace déformé depuis une reconstruction dans une grille. En effet, nous proposons de capturer la déformation induite par système de coordonnées par le *binding* des positions d'une grille régulière englobant le support du champs scalaire. Puis de reconstruire l'interpolation de l'espace et la correspondance de la position inverse linéairement ou quadratiquement.

Dans un premier temps nous avons proposé une détection peu robuste basée sur le plus proche voisin que nous avons amélioré en proposant une détection basée sur une découpe en tétraèdres des hexaèdres. Pour obtenir une reconstruction plus pertinente nous passons par une première phase de reconstruction depuis les coordonnées barycentriques dans les tétraèdres ce qui donne une position de départ pour ensuite lancer la reconstruction numérique de la position inverse dans la base de fonctions choisie.

L'avantage offert par cette détection basée région est de pouvoir détecter des collisions dans l'espace déformé ce qui nous donne par la suite plusieurs antécédents. Cette détection d'antécédents multiples nous a orienté sur l'idée de composer une surface implicite avec elle-même. En particulier, nous nous sommes intéressés à l'utilisation d'opérateurs non associatifs comme l'opérateur de contact.

Ceci permet de proposer des solutions automatiques de résolution du champs scalaire lors d'auto-contacts de l'iso-surface ou d'auto-intersections pour former un pli tout en utilisant des déformations basées cage. Puis, nous nous sommes intéressés à l'idée de proposer l'utilisation de ce champs scalaire avec l'Implicit skinning uniquement avec des déformations basées cage ceci pour corriger des auto-contacts du maillage et des plis formés par la déformation de celui-ci.

15 Perspectives et travaux futurs

Le principal problème posé par notre méthode aux plis est posé par le fait que l'interpolation de l'espace se déforme de sorte créer une partie de l'espace qui se renverse ce qui s'accorde avec la déformation effectuée. Ceci met notre méthode en défaut du fait que le gradient sur la frontière entre la partie de l'espace déformé dans la même orientation que l'espace d'origine et la partie de l'espace renversée devient problématique pour la projection des sommets par l'Implicit skinning. De plus, l'utilisation d'une grille discrète ne permet pas l'entière capture et détection d'un antécédent sur cette frontière. Bien qu'il soit possible d'améliorer la méthode par exemple en étudiant d'autres bases de fonctions ou une résolution multivariée par exemple. Je pense que nous pourrions obtenir de meilleurs résultats en partant sur la piste d'une autre méthode.

Je propose donc de changer de méthode pour un concept qui se baserait sur la capture de la déformation de la surface et non l'interpolation de l'espace. Ceci conduirait à étudier la déformation sous un autre angle. En effet, ici, les cas à étudier relèveraient principalement de la contraction de la surface et son auto-intersection. En partant sur cette idée, nous pouvons espérer avoir une surface implicite déformée compatible avec un système de coordonnées qui n'interpole par l'extérieur de la cage. De plus, nous pouvons espérer avoir une méthode à l'évaluation plus rapide et une meilleure précision sur la surface. Pour pousser le travail plus loin, nous pourrions proposer une surface implicite en champs de distance à échelle variable ce qui permet de conserver une composition intéressante pour des parties plus fines de la surface qu'un champs de distance classique, ainsi que potentiellement proposer des effets de densité de la surface pour la production d'effets physiques en utilisant des opérateurs de grossissement et de tension pour la composition de surfaces implicites.

J'ai l'opportunité de continuer mes travaux de stage vers des résultats qui mèneraient à la publication d'un article de recherche pour décembre, puis ensuite de continuer mes travaux sur cette thématique en thèse sous la supervision de Loïc Barthe, Paul Kry et Brian Wyvill.

Références

[Angelidis and Cani, 2002] Angelidis, A. and Cani, M.-P. (2002). Adaptive implicit modeling using subdivision curves and surfaces as skeletons. In *Symposium on Solid Modeling and Applications*.

[Angles et al., 2017] Angles, B., Tarini, M., Wyvill, B., Barthe, L., and Tagliasacchi, A. (2017). Sketch-based implicit blending. *ACM Trans. Graph.*, 36 :181 :1–181 :13.

[Biasotti et al., 2018] Biasotti, S., Moscoso Thompson, E., Barthe, L., Berretti, S., Giachetti, A., Lejemble, T., Mellado, N., Moustakas, K., Manolas, I., Dimou, D., Tortorici, C., Velasco-Forero, S., Werghi, N., Polig, M., Sorrentino, G., and Hermon, S. (2018). Recognition of Geometric Patterns Over 3D Models. In Telea, A., Theoharis, T., and Veltkamp, R., editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association.

[Bloomenthal and Shoemake, 1991] Bloomenthal, J. and Shoemake, K. (1991). Convolution surfaces. In *SIGGRAPH*.

[Bridson, 2007] Bridson, R. (2007). Fast poisson disk sampling in arbitrary dimensions. In *SIGGRAPH Sketches*.

[Canezin et al., 2013] Canezin, F., Guennebaud, G., and Barthe, L. (2013). Adequate inner bound for geometric modeling with compact field functions. *Computers & Graphics*, 37 :565–573.

[Casti et al., 2019] Casti, S., Livesu, M., Mellado, N., Rumman, N. A., Scateni, R., Barthe, L., and Puppo, E. (2019). Skeleton based cage generation guided by harmonic fields. *Computers & Graphics*, 81 :140–151.

[Floater, 2003] Floater, M. S. (2003). Mean value coordinates.

[Forest et al., 2009] Forest, V., Barthe, L., Guennebaud, G., and Paulin, M. (2009). Soft textured shadow volume. *Computer Graphics Forum, Eurographics Symposium on Rendering 2009*, 28(4) :1111–1121.

[García et al., 2013] García, F. G., Paradinas, T., Coll, N., and Patow, G. (2013). * cages : : A multilevel, multi-cage-based system for mesh deformation. *ACM Transactions on Graphics (TOG)*, 32(3) :24.

[Gourmel et al., 2013] Gourmel, O., Barthe, L., Cani, M.-P., Wyvill, B., Bernhardt, A., Paulin, M., and Grasberger, H. (2013). A Gradient-Based Implicit Blend. *ACM Transactions on Graphics*, 32(2) :1–12.

[Gourmel et al., 2010] Gourmel, O., Pajot, A., Paulin, M., Barthe, L., and Poulin, P. (2010). Fitted BVH for Fast Raytracing of Metaballs. *Computer Graphics Forum, Eurographics 2010 Proceedings*, 29(2) :281–288.

[Hamilton, 1848] Hamilton, W. R. (1848). Xi. on quaternions ; or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 33(219) :58–60.

[Hormann and Floater, 2006] Hormann, K. and Floater, M. S. (2006). Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph.*, 25 :1424–1441.

[Jin and Tai, 2002] Jin, X. and Tai, C.-L. (2002). Convolution surfaces for arcs and quadratic curves with a varying kernel. *The Visual Computer*, 18 :530–546.

[Joshi et al., 2007] Joshi, P., Meyer, M., DeRose, T., Green, B., and Sanocki, T. (2007). Harmonic coordinates for character articulation. *ACM Trans. Graph.*, 26 :71.

[Ju et al., 2005] Ju, T., Schaefer, S., and Warren, J. D. (2005). Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, 24 :561–566.

[Kavan et al., 2007] Kavan, L., Collins, S., Zára, J., and O'Sullivan, C. (2007). Skinning with dual quaternions. In *SI3D*.

[Langer et al., 2006] Langer, T., Belyaev, A., and Seidel, H.-P. (2006). Spherical barycentric coordinates. In *Symposium on Geometry Processing*, pages 81–88.

[Li et al., 2013] Li, X.-Y., Ju, T., and Hu, S.-M. (2013). Cubic mean value coordinates. *ACM Trans. Graph.*, 32 :126 :1–126 :10.

[Lipman et al., 2007] Lipman, Y., Kopf, J., Cohen-Or, D., and Levin, D. (2007). Gpu-assisted positive mean value coordinates for mesh deformations. In *Symposium on Geometry Processing*.

[Lipman et al., 2008] Lipman, Y., Levin, D., and Cohen-Or, D. (2008). Green coordinates. *ACM Trans. Graph.*, 27 :78.

[Macedo et al., 2011] Macedo, I., Gois, J. P., and Velho, L. (2011). Hermite radial basis functions implicits. In *Computer Graphics Forum*, volume 30, pages 27–42. Wiley Online Library.

[Magnenat-Thalmann et al., 1989] Magnenat-Thalmann, N., Laperrière, R., and Thalmann, D. (1989). Joint-dependent local deformations for hand animation and object grasping.

[Mellado et al., 2017] Mellado, N., Vanderhaeghe, D., Hoarau, C., Christophe, S., Br'edif, M., and Barthe, L. (2017). Constrained palette-space exploration. *ACM Trans. Graph.*, 36(4) :60 :1–60 :14.

[Meyer et al., 2002] Meyer, M., Barr, A. H., Lee, H., and Desbrun, M. (2002). Generalized barycentric coordinates on irregular polygons. *J. Graphics, GPU, & Game Tools*, 7 :13–22.

[Miklos et al., 2010] Miklos, B., Giesen, J., and Pauly, M. (2010). Discrete scale axis representations for 3d geometry. In *SIGGRAPH '10*.

[Möbius, 1827] Möbius, A. F. (1827). *Der barycentrische Calcul.* Barth.

[Pajot et al., 2011] Pajot, A., Barthe, L., Paulin, M., and Poulin, P. (2011). Combinatorial Bidirectional Path-Tracing for Efficient Hybrid CPU/GPU Rendering. *Computer Graphics Forum, Eurographics 2011*, 30(2) :315–324. DOI : 10.1111/j.1467-8659.2011.01863.x.

[Parry, 1986] Parry, S. R. (1986). Free-form deformations in a constructive solid geometry modeling system.

[Pasko et al., 1995] Pasko, A. A., Adzhiev, V., Sourin, A., and Savchenko, V. V. (1995). Function representation in geometric modeling : concepts, implementation and applications. *The Visual Computer*, 11 :429–446.

[Requicha, 1980] Requicha, A. A. G. (1980). Representations for rigid solids : Theory, methods, and systems. *ACM Comput. Surv.*, 12 :437–464.

[Roussellet et al., 2018] Roussellet, V., Rumman, N. A., Canezin, F., Mellado, N., Kavan, L., and Barthe, L. (2018). Dynamic implicit muscles for character skinning. *Computers & Graphics*, 77 :227–239.

[Sabin, 1968] Sabin, M. (1968). The use of potential surfaces for numerical geometry. *British Aircraft Corporation, Weybridge, UK, Technical Report No. VTO/MS/153.*

[Savoye, 2017] Savoye, Y. (2017). Stokes coordinates.

[Sederberg and Parry, 1986] Sederberg, T. W. and Parry, S. R. (1986). Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics*, 20(4) :151–160.

[Sorkine-Hornung and Alexa, 2007] Sorkine-Hornung, O. and Alexa, M. (2007). As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing.*

[Tachella et al., 2019] Tachella, J., Altmann, Y., Mellado, N., McCarthy, A., Tobin, R., Buller, G. S., Tourneret, J.-Y., and McLaughlin, S. (2019). Real-time 3d reconstruction of complex scenes using single-photon lidar : when image processing meets computer graphics.

[Vaillant et al., 2013] Vaillant, R., Barthe, L., Guennebaud, G., Cani, M.-P., Rohmer, D., Wyvill, B., Gourmel, O., and Paulin, M. (2013). Implicit skinning : real-time skin deformation with contact modeling. *ACM Trans. Graph.*, 32 :125 :1–125 :12.

[Vaillant et al., 2014] Vaillant, R., Guennebaud, G., Barthe, L., Wyvill, B., and Cani, M.-P. (2014). Robust iso-surface tracking for interactive character skinning. *ACM Trans. Graph.*, 33 :189 :1–189 :11.

[Vergne et al., 2011] Vergne, R., Vanderhaeghe, D., Chen, J., Barla, P., Granier, X., and Schlick, C. (2011). Implicit Brushes for Stylized Line-based Rendering. *Computer Graphics Forum, Eurographics 2011*, 30(2) :513–522.

[Wachspress, 1975] Wachspress, E. L. (1975). *A rational finite element basis.* Elsevier.

[Wang et al., 2015] Wang, Y., Jacobson, A., Barbic, J., and Kavan, L. (2015). Linear subspace design for real-time shape deformation. *ACM Trans. Graph.*, 34 :57 :1–57 :11.

[Weber et al., 2012] Weber, O., Poranne, R., and Gotsman, C. (2012). Biharmonic coordinates. *Comput. Graph. Forum*, 31 :2409–2422.

[Wyvill et al., 1986] Wyvill, G., McPheeters, C., and Wyvill, B. (1986). Data structure for soft objects. *The Visual Computer*, 2 :227–234.

[Zanni, 2013] Zanni, C. (2013). *Skeleton-based implicit modeling and applications.* Theses, Université de Grenoble.

[Zhang et al., 2014] Zhang, J., Deng, B., Liu, Z., Patanè, G., Bouaziz, S., Hormann, K., and Liu, L. (2014). Local barycentric coordinates. *ACM Trans. Graph.*, 33 :188 :1–188 :12.

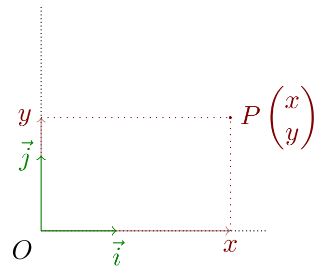
Septième partie
Annexes

A Coordonnées barycentriques

A.1 Coordonnées barycentriques dans les triangles et tétrahèdres

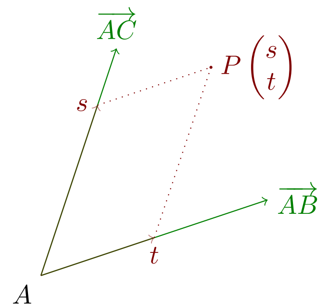
Pour comprendre les coordonnées barycentriques généralisées aux cages de déformation en section7.1, étudions les coordonnées barycentriques définies pour les triangles (2D) et les tétrahèdres (3D). Nous comprenons comment lire les coordonnées cartésiennes d'une position dans un repère orthonormé. Il suffit de regarder le projeté de notre position sur l'axe choisi et lire sa distance à l'origine ce qui correspond à estimer le changement d'échelle demandé au vecteur normé qui définit l'axe pour atteindre le projeté depuis l'origine. Dire qu'une position P est aux coordonnées cartésiennes (x, y) dans un repère $\left(O, \vec{i}, \vec{j}\right)$ revient à dire qu'il peut s'écrire :

$$P = O + x\vec{i} + y\vec{j}$$



Dans le même esprit il est possible de définir un repère quelconque depuis trois positions A, B et C non alignées. De cette manière on définit un repère $\left(A, \overrightarrow{AB}, \overrightarrow{AC}\right)$ dans lequel on peut exprimer toute position P par des coordonnées (s, t) dans ce repère :

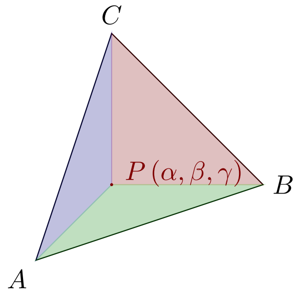
$$P = A + s\overrightarrow{AB} + t\overrightarrow{AC}$$



En réalité cette représentation est équivalente à définir des coordonnées barycentriques, c'est-à-dire :

$$\begin{aligned} P &= A + s(B - A) + t(C - A) \\ P &= (1 - s - t)A + sB + tC \\ P &= \alpha A + \beta B + \gamma C \end{aligned}$$

Ces coordonnées (α, β, γ) sont les coordonnées barycentriques de P dans le triangle (ABC) et sont telles que $\alpha + \beta + \gamma = 1$ $(= (1 - s - t) + s + t)$.



Pour trouver les coordonnées barycentriques depuis les coordonnées cartésiennes, il est possible de le poser comme un système linéaire :

$$\begin{array}{rcl} P & = & (1-s-t)A + sB + tC \\ P - A & = & s\,(B-A) + t\,(C-A) \\ \overrightarrow{AP} & = & s\overrightarrow{AB} + t\overrightarrow{AC} \\ \overrightarrow{AP} & = & \left(\overrightarrow{AB}, \overrightarrow{AC}\right)\begin{pmatrix} s \\ t \end{pmatrix} \\ \left(\overrightarrow{AB}, \overrightarrow{AC}\right)^{-1}\overrightarrow{AP} & = & \begin{pmatrix} s \\ t \end{pmatrix}. \end{array}$$

De plus, $P \in (ABC)$ est équivalent à dire que $(\alpha, \beta, \gamma) = (1-s-t, s, t) \in [0,1]^3$.

De même, on peut résoudre ce système par la méthode de Cramer dont la représentation a un sens ici. En effet, le déterminant d'une matrice 2×2 correspond à l'aire du parallélogramme induit par les deux vecteurs qui forment la matrice. Ceci fait par conséquent que :

$$\beta = s = \frac{\det\left(\overrightarrow{AP}, \overrightarrow{AC}\right)}{\det\left(\overrightarrow{AB}, \overrightarrow{AC}\right)} = \frac{\texttt{aire}(APC)}{\texttt{aire}(ABC)}.$$

En effet, les coordonnées barycentriques correspond au ratio d'aire signée que le triangle opposé au sommet représente dans le triangle qui sert de repère. Ceci se vérifie aussi pour

$$\gamma = t = \frac{\det\left(\overrightarrow{AB}, \overrightarrow{AP}\right)}{\det\left(\overrightarrow{AB}, \overrightarrow{AC}\right)}$$

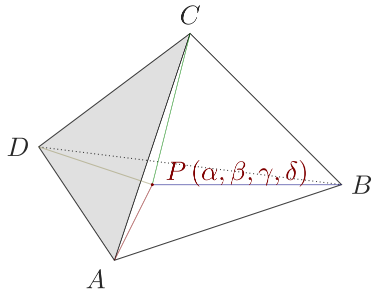
et

$$\alpha = 1 - s - t = \frac{\det\left(\overrightarrow{AB}, \overrightarrow{AC}\right) - \det\left(\overrightarrow{AP}, \overrightarrow{AC}\right) - \det\left(\overrightarrow{AB}, \overrightarrow{AP}\right)}{\det\left(\overrightarrow{AB}, \overrightarrow{AC}\right)} = \frac{\det\left(\overrightarrow{BP}, \overrightarrow{BC}\right)}{\det\left(\overrightarrow{BA}, \overrightarrow{BC}\right)}.$$

À noter que le centre de masse est la position d'équilibre qui partitionne un triangle en 3 aires égales ($\alpha = \beta = \gamma = \frac{1}{3}$).

Les coordonnées barycentriques dans les triangles (2D) se généralisent en 3D dans les tétrahèdres. En effet pour un repère définit par un tétrahèdre ($ABCD$) non plat : $\left(A, \overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{AD}\right)$, on a pour une position P dans l'espace 3D qu'il existe (s, t, u) tels que :

$$\begin{array}{rcl} P & = & A + s\overrightarrow{AB} + t\overrightarrow{AC} + u\overrightarrow{AD} \\ P & = & (1-s-t-u)A + sB + tC + uD \\ P & = & \alpha A + \beta B + \gamma C + \delta D. \end{array}$$



Il est possible de retrouver les coordonnées barycentriques $(\alpha, \beta, \gamma, \delta) = (1 - s - t - u, s, t, u)$ dans $(ABCD)$ depuis les coordonnées cartésiennes par une résolution de système linéaire comme présenté pour les triangles :

$$\begin{pmatrix} s \\ t \\ u \end{pmatrix} = \left(\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{AD}\right)^{-1} \overrightarrow{AP}.$$

La remarque concernant les ratios d'aire pour les triangles se généralise en admettant les ratios de volumes pour les tétrahèdres car le déterminant d'une base de 3 vecteurs en 3D donne le volume du parallélépipède qu'ils impliquent.

De plus les exemples des triangles et tétrahèdres sont intéressants pour voir l'intérêt des coordonnées barycentriques pour la déformation. En effet, elles proposent une interpolation linéaire sur le triangle, ce qui est utilisé par exemple pour afficher une texture sur les triangles d'un maillage depuis ses coordonnées de textures peu importe la déformation du triangle.

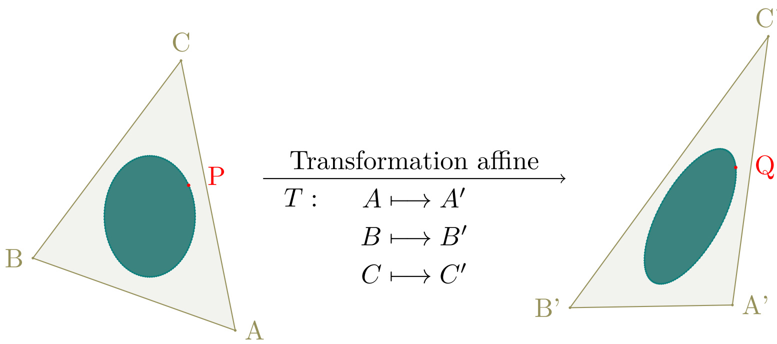


FIGURE 67 – **Déformation d'un disque par une cage triangulaire**
P est exprimé dans le triangle (ABC) par ses coordonnées barycentriques (α, β, γ) : $P = \alpha A + \beta B + \gamma C$. La transformation T qui transforme (ABC) en $(A'B'C')$: c'est équivalent à un changement de repère. Ceci donne la correspondance de P dans $(A'B'C')$ comme étant la position $Q = \alpha A' + \beta B' + \gamma C'$.

Si l'on souhaite envoyer une position $P \in (ABC)$ dans un triangle $(A'B'C')$, ceci fonctionne comme un changement de repère, en effet le repère $\left(A, \overrightarrow{AB}, \overrightarrow{AC}\right)$ définit un espace affine d'origine A, dans lequel on peut exprimer P par des coordonnées (s, t) qui donnent la combinaison linéaire de $\overrightarrow{AB}$ et $\overrightarrow{AC}$ permettant d'obtenir P dans $\left(A, \overrightarrow{AB}, \overrightarrow{AC}\right)$. Pour changer de repère, il est possible d'utiliser ces coordonnées pour lire un équivalent Q dans le repère $\left(A', \overrightarrow{A'B'}, \overrightarrow{A'C'}\right)$: $Q = A' + s\overrightarrow{A'B'} + t\overrightarrow{A'C'} = \alpha A' + \beta B' + \gamma C'$ comme illustré en figure 67. On remarque ici que ce changement de repère exprime $P \in (ABC)$ et $Q \in (A'B'C')$ par les mêmes coordonnées barycentriques. En effet, la déformation de (ABC) en $(A'B'C')$ définit une transformation affine de l'espace exprimé dans le repère $\left(A, \overrightarrow{AB}, \overrightarrow{AC}\right)$ vers le l'espace définit par $\left(A', \overrightarrow{A'B'}, \overrightarrow{A'C'}\right)$. Cette propriété est l'invariance affine des coordonnées barycentriques.

A.2 Invariance affine des coordonnées barycentriques généralisées

Nous établissons ici le fait qu'une transformation affine est linéaire par rapport aux coordonnées barycentriques dans une cage de déformation quelconque.

Soit $T : C_i \longmapsto D_i$ une transformation affine qui envoie l'espace affine $C_0 \times \texttt{vect}\left(\overrightarrow{C_0C_1}, \ldots, \overrightarrow{C_0C_n}\right)$ vers $D_0 \times \texttt{vect}\left(\overrightarrow{D_0D_1}, \ldots, \overrightarrow{D_0D_n}\right)$ et de partie linéaire $\overrightarrow{T}$. Prenons pour convention que pour deux positions A et B, on peut écrire $\overrightarrow{AB} = B - A$. De plus $\overrightarrow{T}$ est linéaire par rapport aux vecteurs et $\overrightarrow{T}\left(\overrightarrow{AB}\right) = \overrightarrow{T(A)\,T(B)}$ par définition de la transformation affine. Ceci nous donne que pour une position P et ses coordonnées barycentriques $\{\alpha_i(P)\}_{i\in[n]}$:

$$P = \textstyle\sum_{i\in[n]} \alpha_i(P) C_i,$$

car les $\{\alpha_i(P)\}_{i\in[n]}$ sont des coordonnées barycentriques (donc de somme 1) :

$$\begin{aligned}
\left(\sum_{i\in[n]} \alpha_i(P)\right) P &= \sum_{i\in[n]} \alpha_i(P) C_i,\\
\sum_{i\in[n]} \alpha_i(P)\,(P - C_i) &= \vec{0},\\
\sum_{i\in[n]} \alpha_i(P)\left(\overrightarrow{C_iP}\right) &= \vec{0},\\
\overrightarrow{T}\left(\sum_{i\in[n]} \alpha_i(P)\left(\overrightarrow{C_iP}\right)\right) &= \overrightarrow{T}\left(\vec{0}\right),
\end{aligned}$$

par linéarité de $\overrightarrow{T}$:

$$\sum_{i\in[n]} \alpha_i(P)\overrightarrow{T}\left(\overrightarrow{C_iP}\right) = \vec{0},$$

car T est affine de partie linéaire $\overrightarrow{T}$:

$$\begin{aligned}
\sum_{i\in[n]} \alpha_i(P)\overrightarrow{T(C_i)\,T(P)} &= \vec{0},\\
\sum_{i\in[n]} \alpha_i(P)\,(T(P) - T(C_i)) &= \vec{0},\\
\left(\sum_{i\in[n]} \alpha_i(P)\right) T(P) &= \sum_{i\in[n]} \alpha_i(P) T(C_i),\\
T(P) &= \sum_{i\in[n]} \alpha_i(P) D_i.
\end{aligned}$$

Une transformation affine est donc linéaire par rapport aux coordonnées barycentriques

□

B Problèmes de continuité de la première approche : résolution directe

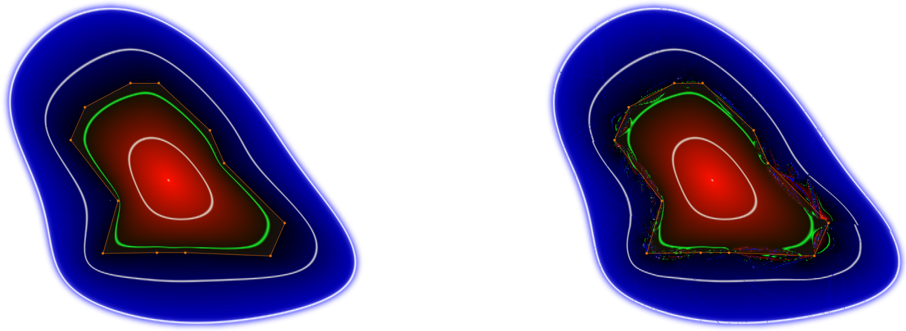
B.1 Problème de continuité des dérivées partielles au bord de la cage

Nous justifions ici l'utilisation en section 8.2 de la matrice Jacobienne fabriquée depuis les différences finies en tant que dérivées partielles au lieu de la matrice Jacobienne obtenue de manière calculatoire.

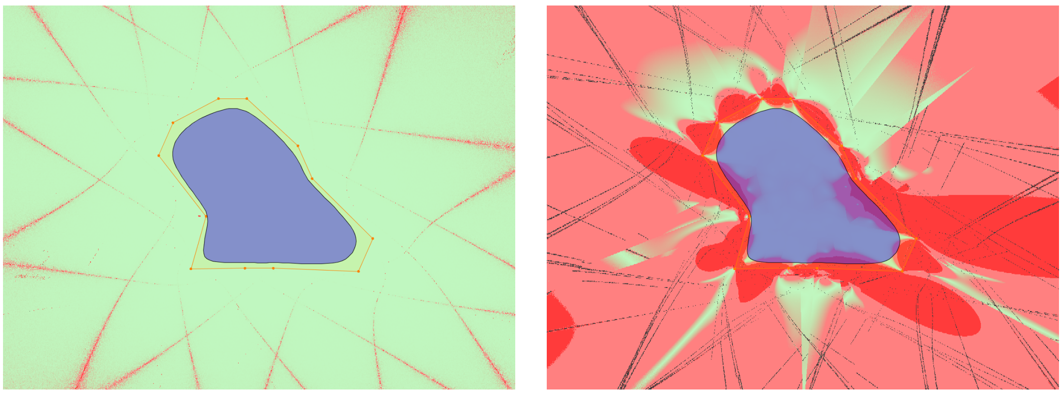
Je propose de mesurer l'erreur de notre méthode par la distance suivante :

$$\|Q - T(P(Q))\|$$

où $P(Q)$ est la position inverse trouvée pour Q. L'intérêt de cette erreur est de vérifier que l'inverse se transforme en la position que nous cherchions à inverser.



(a) Différences finies (b) Dérivées partielles



(c) Différences finies (erreur) (d) Dérivées partielles (erreur)

FIGURE 68 – **Problèmes de continuité des dérivées au bord de la cage**
En (a) le champ scalaire en utilisant la matrice Jacobienne depuis les différences finies tel que proposé en section 8.2 et (b) la matrice Jacobienne calculée depuis le système de coordonnées barycentriques Mean Value Coordinates. En (c) et (d) en rouge du vert au rouge clair les erreurs de 0 *à* 10^{-2}*, en rouge clair entre* 10^{-2} *et* $5 \cdot 10^{-2}$ *et en rouge vif au dessus de* $5 \cdot 10^{-2}$*. Nous observons en particulier que pour (b) (d) la méthode ne converge pas au bord de la cage et ne permet pas de trouver une position inverse.*

En figure 68 nous illustrons les résultats obtenus par notre méthode et le problème que pose l'utilisation de la matrice Jacobienne par le calcul des dérivées partielles. Le calcul des *Mean Value Coordinates* en 2D est donné par :

$$a_i = C_i - P,\ b_i = C_{i+1} - P,$$

$$\delta_i = \frac{\det(a_i, b_i)}{|\det(a_i, b_i)|} \frac{a_i \cdot b_i}{||a_i||||b_i||}$$

$$\omega_i = \frac{\tan\left(\frac{\delta_{i-1}}{2}\right) + \tan\left(\frac{\delta_i}{2}\right)}{||a_i||},$$

$$P = \frac{\sum\limits_{i\in[n]} \omega_i C_i}{\sum\limits_{i\in[n]} \omega_i}$$

Ce qui conduit à la dérivation suivante :

$$\frac{\partial}{\partial\gamma}\delta_i = \frac{\partial}{\partial\gamma}\frac{a_i \cdot b_i}{||a_i||||b_i||} = (a_i \cdot b_i)\frac{a_{i,\gamma}||b_i||^2 + b_{i,\gamma}||a_i||^2}{||a_i||^3||b_i||^3} - \frac{a_{i,\gamma} + b_{i,\gamma}}{||a_i||||b_i||}$$

$$\frac{\partial}{\partial\gamma}\tan\left(\frac{\delta_i}{2}\right) = \frac{1 + \tan^2\left(\frac{\delta_i}{2}\right)}{2}\sqrt{\frac{||a_i||||b_i||}{||a_i||||b_i|| - a_i \cdot b_i}}\frac{\partial}{\partial\gamma}\delta_i\frac{\det(a_i, b_i)}{|\det(a_i, b_i)|}$$

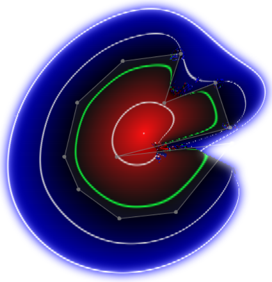
$$\frac{\partial}{\partial\gamma}\omega_i = \frac{\left(\frac{\partial}{\partial\gamma}\tan\left(\frac{\delta_{i-1}}{2}\right) + \frac{\partial}{\partial\gamma}\tan\left(\frac{\delta_i}{2}\right)\right)||a_i|| + \frac{a_{i,\gamma}}{||a_i||}\left(\tan\left(\frac{\delta_{i-1}}{2}\right) + \tan\left(\frac{\delta_i}{2}\right)\right)}{||a_i||^2}$$

$$\frac{\partial}{\partial\gamma}T(P) = \frac{\left(\sum \frac{\partial}{\partial\gamma}\omega_i T(C_i)\right)\left(\sum \omega_i\right) - \left(\sum \omega_i T(C_i)\right)\left(\sum \frac{\partial}{\partial\gamma}\omega_i\right)}{\left(\sum \omega_i\right)^2}$$

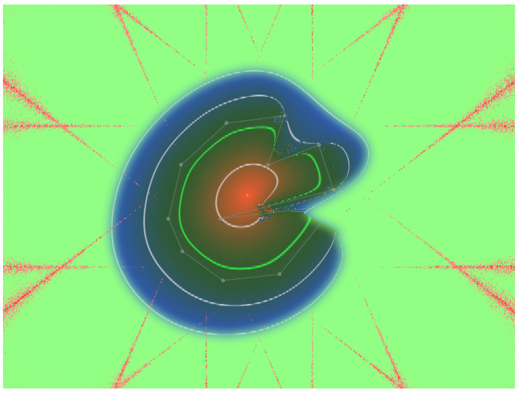
Dans le cas du calcul des coordonnées, il est possible de prolonger la forme close par le calcul des coordonnées barycentriques sur l'arête (interpolation linéaire entre deux points) ce qui permet d'utiliser les différences finies malgré qu'on ne soit que $\mathcal{C}^0$ au bord de la cage. Mais, dans le cas des dérivées partielles, ceci fait apparaître un la quantité $\tan^2\left(\frac{\delta_i}{2}\right)$ et pour un sommet au bord de la cage, on a que la quantité δ_i vaut π (c'est l'angle $(\widehat{C_i P C_{i+1}})$). Or, $\lim\limits_{\delta_i\to\pi}\tan^2\left(\frac{\delta_i}{2}\right) = \infty$ d'où les problèmes de convergence lorsque la position s'approche de la cage.

B.2 Interférences lors de la création de concavités par la déformation

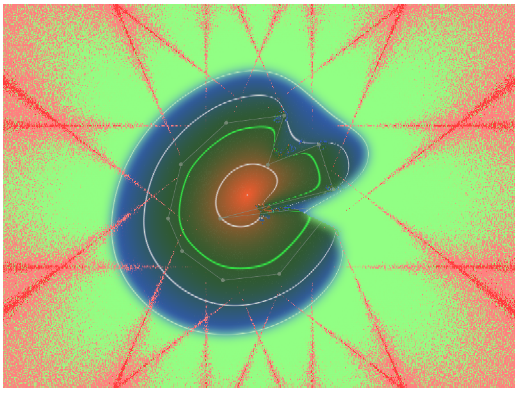
Nous utilisons la mesure d'erreur proposée en section B.1 pour analyser ce qu'il se passe à la création de concavités. En général 5 itérations sont acceptables pour obtenir une erreur inférieure à 10^{-2} pixels. Cependant la déformation par cette méthode est peu robuste à la création de concavités dans la cage. Ceci crée des interférences dans le champ déformé (figure 69). Ce problème est celui des antécédents multiples lors des repliements et contractions de l'espace, dont une explication est donnée en section 9.3 (je conseille au lecteur de ne pas se rendre directement à cette section sans lire les parties qui précèdent).



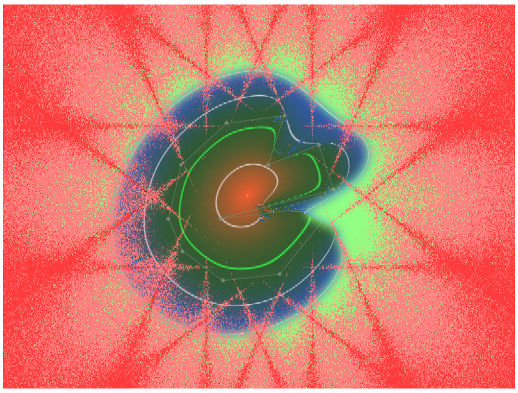
(a) Champ déformé



(b) Erreur (10^{-2})



(c) Erreur (10^{-3})



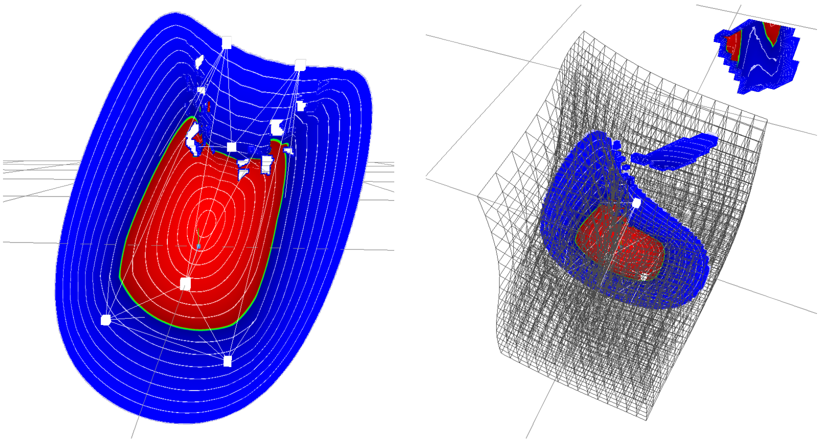
(d) Erreur (10^{-4})

FIGURE 69 – **Problèmes de résolution du champ aux concavités**
En (a) nous observons un problème d'interférence dans les valeurs du champ lors de la création de concavités de la cage. En (b) le seuil d'erreur est donné à 10^{-2}. *En (c) pour une erreur à* 10^{-3} *et (d)* 10^{-4}. *On observe ici que les interférences aux concavités ne sont pas considérées comme des erreurs. En effet, elles correspondent à un antécédent valide pour la transformation.*

C Problèmes de détection des hexaèdres

C.1 Problème de détection par le plus proche voisin et produit vectoriel

Ici nous étudions le manque de robustesse de la méthode présentée en section 9.2.1. Cette analyse a pour but de comprendre ce qu'il se passe et l'intérêt de la solution apportée ensuite.



(a) Interférences (b) Champ erroné

FIGURE 70 – **Problème de détection par le plus proche voisin et produit vectoriel**
En (a), la méthode est peu robuste aux écrasements des hexaèdres, leur retournement et aussi aux antécédents multiples ce qui crée les interférences à la création d'une concavité importante. En (b), le problème mis en évidence est l'apparition de valeurs scalaires non-nulles hors du support et de la grille.

La principale faiblesse de cette méthode est qu'elle manque de robustesse par exemple lors de déformations qui entraînent une compression de l'espace comme illustré par les résultats en figure 70. Ce champ erroné existe car la position cherche un plus proche voisin qui existe toujours, sauf qu'ici l'hexaèdre est mal orienté ce qui trompe la détection par le produit scalaire. Ceci indique que la position appartient à un hexaèdre et lance la reconstruction : la forme de l'hexaèdre déformé permet malgré tout de trouver une position inverse dans le support de la surface d'origine ce qui fait apparaître un champ erroné à l'extérieur de la grille.

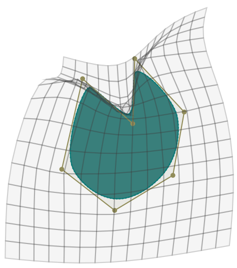
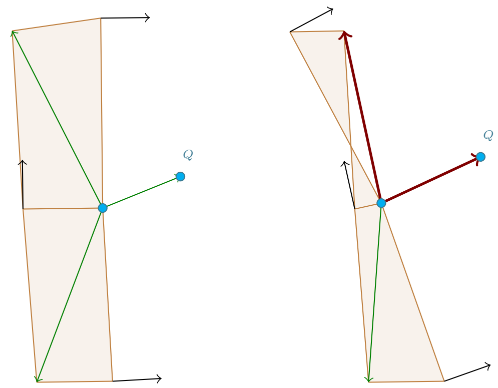


FIGURE 71 – **Compression de l'espace lors de la création d'une concavité importante dans la cage**

Lors d'une compression de l'espace (figure 71), la déformation induite par la cage et le système de coordonnées fait que lorsque cette déformation est trop importante et fait apparaître une concavité, l'espace interpolant capturé par la grille va se compresser ce qui entraînement un aplatissement des cellules mais aussi se plier comme le ferait un tissu ce qui laisse apparaître une portion où les cellules sont renversées pour suivre le changement d'orientation de la déformation.



(a) Détection correcte (b) Détection biaisée

Figure 72 – **Produit scalaire trompé par un retournement d'arête**
En (a), la détection telle qu'on l'attend et qui se produit lors de déformation simples. Des déformations plus complexes de l'espace peut amener au retournement d'arêtes (b) en particulier au bord de la cage. Ceci amène la méthode à valider un hexaèdre qui n'englobe pas la position évaluée par l'erreur de détection de l'orientation de la diagonale.

Ceci entraîne le retournement de certaines arêtes (figure 72). Ces problème font qu'il est possible de ne pas faire la reconstruction dans la bonne cellule. Mais surtout que ceci peut faire apparaître un champ hors de la grille (figure 70).

D Proposition de troisième approche : reconstruction d'un champ de distance

D.1 Capture de la déformation et reconstruction du champ scalaire

La méthode que je souhaite proposer se base sur l'idée de capture la déformation de la surface uniquement. Notre échantillonnage serait construit depuis la subdivision de la cage : par exemple subdiviser chaque triangle depuis ses coordonnées barycentriques par une résolution r en fait r^2 triangles. Ensuite, l'idée est d'utiliser l'*Hermite Radial Basis Functions* ou le champ de distance signé utilisé pour construire la surface implicite et y projeter notre cage subdivisée sur l'iso-surface. Ceci fait, le *binding* sera effectué pour les sommets de ce maillage que je noterai par la suite le *maillage articulé* qui fera office de surface implicite. Ici, nous ne prenons pas en compte le maillage, mais uniquement la cage et la surface implicite.

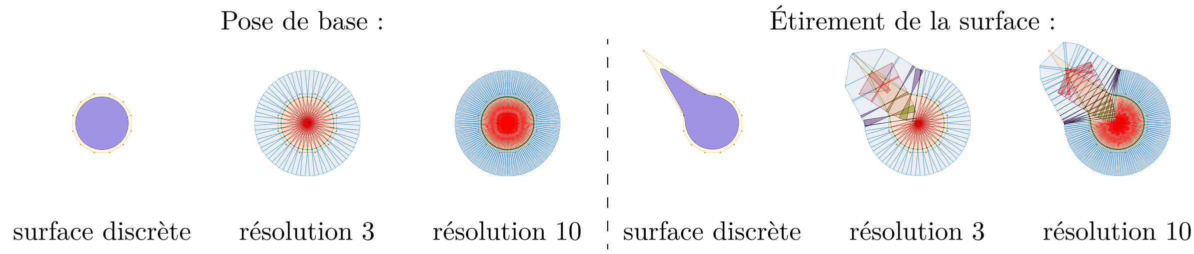


FIGURE 73 – **Représentation d'une surface articulée en 2D**

Une fois le *maillage articulé* défini, je propose de ne plus prendre en compte la surface d'origine ce qui ne nous demande plus de résoudre le problème de la position inverse. En section D.3 je proposerai des concepts auxquels ceci laisse place. Mais, un des intérêt est de potentiellement pouvoir gérer des surfaces concaves. En effet, si une surface d'origine était concave et que cette concavité est perdue par la déformation, ceci pourrait entraîner un étirement du champ et potentiellement un manque de pertinence en particulier pour des configurations telles que les jambes d'un personnage. Je propose donc de baser notre reconstruction depuis les normales en chaque triangle comme illustré en 2D sur la figure 73. Depuis cette normale, l'idée est de construire un prisme triangulaire. La détection pouvant se faire par des produits scalaire et l'estimation de la distance par un produit scalaire.

D.2 Détection multiple

Ces prismes vont être liés aux arêtes par des prismes triangulaires (sur le côté) et en chaque sommet par un ensemble de tétraèdres. Cette partie de la géométrie sera nommé les *liaisons*. Nous supposons que les tétraèdres sont définis par 3 axes de même longueur. L'idée est de mettre à l'échelle ces formes de sorte de capture tout le champ de distance non nul lors de la détection. Dans le cas où un tétraèdre ou un prisme est trop écrasé et donc sa boîte englobante trop grande, ils peuvent être remplacés respectivement par un prisme et un hexaèdre à faces planes pour la détection. Ensuite l'idée est de reconstruire le champs de distance comme la distance au sommet ou à l'arête de notre échantillonnage.

Ici, la détection est différente de celle proposée pour détecter les auto-intersections de l'espace interpolant avec lui-même. C'est par l'aspect des liaisons que nous allons gérer les cas de détections multiples (figure 74) :

— Contraction de la surface : ici c'est la création d'une concavité, ceci ne demande par l'utilisation d'un opérateur de contact, donc à l'extérieur ceci sera résolu par un opérateur max et à l'intérieur par un opérateur min. Ceci permet de conserver un champ de distance aux concavités et éviter les contractions de l'espace comme observé en section 11.2.

— Auto-intersection et auto-contact : lors de la création d'une boucle, ceci ne crée pas de contraction de la surface (en 2D c'est le changement du *turning number* de la courbe) ce qui peut permettre de gérer ce cas comme un contact.

Cette partie reste à étudier plus en détail par la suite. De plus, il peut être intéressant d'étudier ici des opérateurs n-aires, aussi bien pour gérer des contacts multiples que pour gérer les interactions entre les cas de contractions et de contacts.

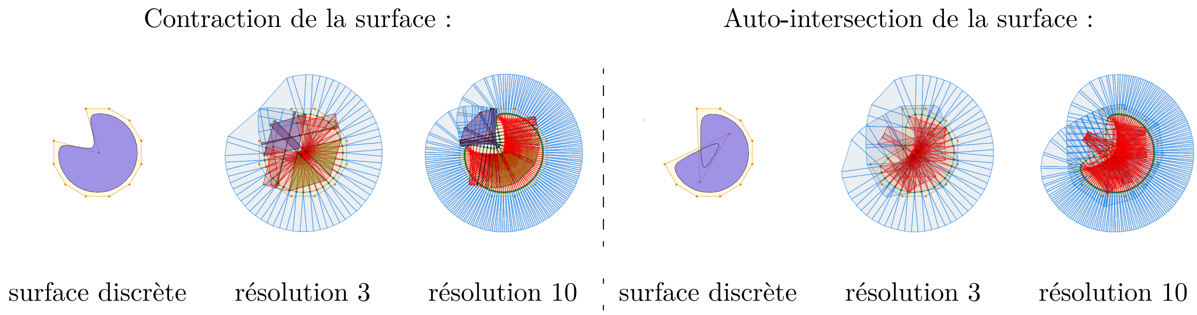


FIGURE 74 – **Détection multiples pour une surface articulée en 2D**
En violet, les liaisons détectent une contraction de l'espace. Dans le cas d'une auto-intersection, ceci n'est pas détecté comme une contraction ce qui permet la résolution au titre d'un auto-contact.

D.3 Reconstruction d'un champ de distance à échelles locales variables

L'intérêt de définir une surface implicite déformable par notre *maillage articulé* est de pouvoir jouer sur les changements d'échelle du champ de distance. De plus, ce changement d'échelle est connu ce qui permet de déterminer un gradient pour effectuer une projection en 1 coup. La mise à l'échelle peut être globale pour proposer un champ de distance qui s'auto-adapte à la déformation en minimisant le squelette topologique mais qu'il existe pour garder un champ intéressant (figure 75).

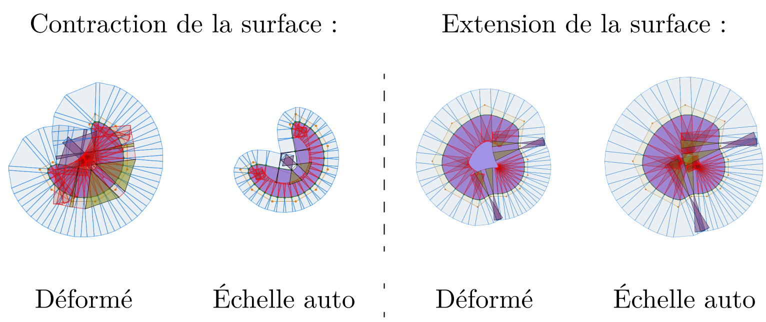


FIGURE 75 – **Mise à l'échelle globale automatique**

De même que le champ de distance peut être défini de sorte de proposer un champ plus intéressant pour la composition qu'un champ de distance classique où les mises à l'échelles peuvent être locales. Par exemple, elles peuvent se baser sur un calcul des axes médians [Miklos et al., 2010] à l'étape de *binding* ce qui peut permettre une composition plus intéressante dans des régions telles que les membres d'un personnage comparé à son corps comme illustré par le *champ de distance multi-échelle* en figure 76. Ici, il peut être intéressant de s'intéresser à définir une seule surface implicite pour une entité contrôlée par une cage. De plus, il est peut être intéressant de proposer l'échelle variable pour proposer des effets de densité (par exemple un squelette topologique plus large pour une densité plus importante) que l'utilisateur pourrait potentiellement paramétrer.

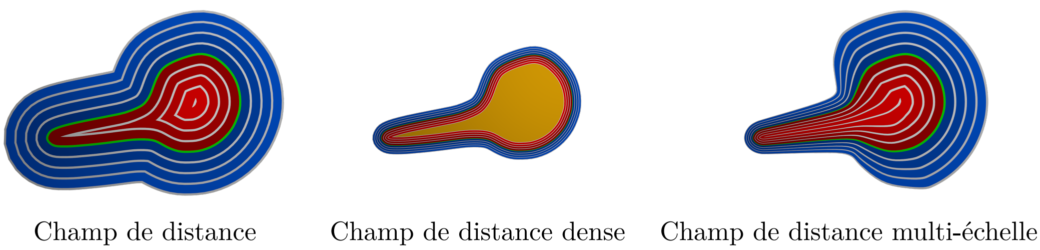


FIGURE 76 – **Champ de distance à échelle variable**

Résumé

Pour satisfaire des attentes toujours plus exigeantes en image de synthèse, la recherche en infographie propose des méthodes de déformations de formes toujours plus réalistes pour donner vie à nos personnages. Bien que des méthodes comme le *Linear Bending Skinning* ou le *Dual Quaternion Skinning* soient populaire dans l'industrie, en 2013 l'*Implicit skinning* est né. C'est une méthode de déformation d'un maillage basée squelette qui permet la conservation de volumes intéressants et la correction automatique propre d'un pli. Cette méthode se base sur la théorie des surfaces implicites et leur composition pour proposer la correction du maillage. Mais cette méthode reste basée sur des déformations rigides contrôlées par un squelette d'animation.

Cependant des déformations basées cages permettent des déformations d'un maillage plus libres et plus souples. Nous aimerions utiliser ces déformations par les cages sur des surfaces implicites pour améliorer la méthode de l'Implicit skinning. Pour proposer ces déformations, la piste proposée dans ce document est la résolution du problème de la position inverse. Dans un premier temps nous présentons les problèmes rencontrés par la résolution numérique directe de ce problème. Puis nous proposons une méthode qui se base sur l'approximation de notre déformation pour reconstruire la position inverse.

Grâce à la détection de collisions depuis une décomposition en tétraèdres de notre espace, nous proposons d'utiliser la surjectivité de notre transformation de l'espace pour proposer la résolution d'antécédents multiples. La résolution d'antécédents multiples nous a permis de proposer une surface implicite dont le champs scalaire résout automatiquement les auto-intersections par une composition depuis l'opérateur de contact utilisé pour l'Implicit skinning.

Ceci nous a conduit à initier des travaux sur la correction automatique d'auto-intersections d'un maillage déformé par une cage en utilisant l'Implicit skinning depuis une seule surface implicite.

Abstract

To satisfy always higher expectations in image synthesis, research in computer graphics propose shape deforming methods always more realistic to make our caracters alive. Even though methods like the *Linear Bending Skinning* or the *Dual Quaternion Skinning* are popular in the industry, in 2013 the *Implicit skinning* was born. It is a skeletal based method to deform a mesh which allow to preserve interesting volumes and the automatic correcting of a clean fold. This method is based on the implicit surfaces theory and their composition to propose a mesh correcting. But this method is only based on rigid deformations directed by an animation skeleton.

However cage based deformations allow a more free and smoother deforming of a mesh. We would like to use these cage based deformations on implicit surfaces to improve the Implicit skinning method. To propose these deformations, the idea proposed by this document is the solving of the inverse position problem. First, we present the problems we encountered by the direct numerical solving of this problem. Then we propose a method based of the approximation of our deformation to reconstruct the inverse position.

Thanks to the collisions detecting using a tetrahedral cut of our space, we propose to use surjectivity of our space mapping to propose the multiple antecedents solving. The multiple antecedents solving let us to propose an implicit surface whose scalar field automatically solve self-intersections using a composition with the contact operator used by the Implicit skinning.

This made us to start our work on the automatic correcting of self-intersections of a cage based deformed mesh by using the Implicit skinning with only one implicit surface.