On Some Interactive Mesh Deformations

Universitat Politècnica de Catalunya

PhD Programme in Computing

Doctoral Thesis

Maria dels Àngels Cerveró Abelló

Advisors: Pere Brunet Crosa and Àlvar Vinacua Pla

December, 2015

On Some Interactive Mesh Deformations

Maria dels Àngels Cerveró Abelló

A thesis submitted in fulfillment for the degree of
Philosophiæ Doctor in Computer Science

PhD Program in Computing
Universitat Politècnica de Catalunya

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Advisors
Prof. Pere Brunet Crosa and Prof. Àlvar Vinacua Pla

December 2015

Abstract

Techniques devoted to deform 3D models are an important research field in Computer Graphics. They can be used in different stages: the modelling phase, the animation process and also during some special simulations. Additionally, some applications may require the manipulation of 3D models under certain restrictions to preserve the volume of the modified object. Hence, the present PhD Dissertation explores new algorithms to perform flexible, robust and efficient 3D deformations. Apart from this, it also researches on a new methodology to restrict these deformations so that the volume of the manipulated model remains constant.

Some of the most used methods to achieve smooth deformations are those included in the *Cage-Based Deformation* paradigm. Cage-based deformations enclose the model to be deformed in a coarse polyhedron, the *cage*. Then, they usually rely on *Generalised Barycentric Coordinates* to relate the model with the vertices, and other geometric elements, of this cage, which are the control points or the deformation handles. Finally, every time that one of these handles is dragged, the model is deformed accordingly. Although this paradigm is simple, elegant and performs efficient deformations, some *cage-free* space deformation techniques have recently appeared. They increase the flexibility of the deformation handles, which do not need to be connected, and define powerful tools that make the deformation process more versatile and intuitive. In this context, the Dissertation introduces new Generalised Barycentric Coordinate systems specially designed to be used in a cage-free environment. Any user who wants to use the presented schemes only needs to locate a set of control points in the vicinity of the model that he or she wants to deform. These handles can be placed wherever he or she considers mode suitable and the only requirement is that the model has to be enclosed in their convex hull.

Up to now, there are few techniques to produce volume-preserving space deformations. However, in recent years there has been a growing interest in performing constrained deformations due to their more realistic and physically plausible results. Our contribution to this research line consists in a deformation framework that preserves the volume of the 3D models by means of its gradient and a control surface to restrict the movement of the handles. Moreover, the proposed methodology is not restricted to the cage-based schemes, but it can also be used in a cage-free environment.

Finally, our research can be specially useful for spatial deformations of biological and medical models. This kind of models represent real organs and tissues, which are often soft and lack an internal rigid structure. In addition, they are elastic and incompressible. Any application designed to deal with this group of models and to train or assist doctors must be flexible, robust, efficient and user-friendly. The combination of the proposed cage-free systems with the presented volume-preserving deformation framework satisfies all these conditions. Hence, our contributions can become powerful tools to produce medical-oriented deformations.

Sinopsi

Les deformacions de models 3D s'utilitzen en diverses etapes de la generació de continguts digitals: durant la fase de modelatge, durant el procés d'animació i en alguns tipus de simulacions. A més a més, hi ha aplicacions que necessiten que la manipulació dels models 3D es faci tenint en compte certes restriccions que permeten la conservació del volum de l'objecte modificat. Tot plegat fa que les tècniques de deformació 3D siguin un camp d'estudi molt important dins del món dels Gràfics. Per aquesta raó, aquesta Tesi Doctoral estudia nous algorismes que permetin realitzar deformacions 3D de manera flexible, robusta i eficient i que, a més a més, permetin conservar el volum dels objectes modificats.

Un dels paradigmes més utilitzats per tal de realitzar deformacions suaus és el conegut amb el nom de *Deformacions Basades en un Poliedre Englobant*. Aquesta família de mètodes embolcalla el model que es vol deformar, normalment representat com una malla de triangles, dins d'un poliedre simple, amb poques cares. Un cop fet això, estableix un sistema de *Coordenades Baricèntriques Generalitzades* per tal de definir els vèrtexs del model a partir dels vèrtexs del poliedre englobant, els quals s'anomenen *punts de control* o *controls de la deformació*. D'aquesta manera, cada cop que s'arrossega o es modifica un d'aquests punts de control, el model que es troba dins del poliedre englobant es deforma segons el sistema de coordenades que s'ha definit. Tot i que aquest paradigma és simple, elegant i eficient, des de fa ja uns anys han començat a aparèixer noves tècniques que no necessiten el poliedre englobant per tal de realitzar la deformació. El seu principal objectiu és augmentar la flexibilitat dels controls de la deformació i definir eines que facin que el procés de deformació sigui més versàtil i intuïtiu. Tenint en compte aquest factor, aquesta Tesi també estudia sistemes de Coordenades Baricèntriques Generalitzades dissenyats per realitzar deformacions sense la necessitat de definir el poliedre englobant. D'aquesta manera, qualsevol usuari que vulgui utilitzar els mètodes que es presenten en aquesta Dissertació només s'ha d'encarregar de definir un conjunt de punts de control al voltant del model que vol deformar, podent-los posar allí on consideri més oportú segons la deformació que vulgui obtenir. L'únic requeriment necessari és que el model ha de quedar dins de l'envolupant convexa d'aquests punts de control.

Actualment existeixen pocs mètodes que realitzin deformacions 3D amb preservació del volum. No obstant això, d'un temps ençà ha augmentat l'interès per realitzar deformacions subjectes a certes restriccions que fan que el resultat sigui més realista i físicament versemblant. La contribució d'aquesta Tesi dins d'aquesta línia de recerca consisteix en un sistema de deformació que preserva el volum dels objectes 3D gràcies a còmput del seu gradient i a una superfície de control que restringeix el moviment dels punts de control. Aquest mètode es pot aplicar tant als sistemes de deformació que necessiten un poliedre englobant com als que no el necessiten.

Finalment, i ja per acabar, la recerca realitzada pot ser especialment útil per tal de realitzar deformacions de models mèdics i biològics. Aquests tipus de models poden representar òrgans i teixits reals, els quals, normalment, són tous, mancats d'una estructura rígida interna, elàstics i incompressibles. Qualsevol aplicació dissenyada per treballar amb aquest tipus de models i per entrenar i donar assistència a usuaris mèdics hauria de ser flexible, robusta, eficient i fàcil d'utilitzar.

La combinació dels mètodes de deformació proposats conjuntament amb el sistema de preservació de volum satisfà totes aquestes condicions. Per aquesta raó es creu que les contribucions realitzades poden esdevenir eines importants per produir deformacions mèdiques.

Acknowledgments

I would like to thank my advisors Pere Brunet and Àlvar Vinacua for their guidance and tutoring during the course of my doctoral studies. I admire them for their wide knowledge and for being my source of learning. I have had their technical and intellectual support all along these years and my doctoral thesis would have never been materialised without their invaluable help.

I want to thank the MOVING Research Group all the material and personal resources at my disposal. In particular, I would like to thank Marta Fairen for all her help and advice during the period when I taugh the lab sessions on the subject of *Visualització i Interacció Gràfica*. That was a wonderful and enjoyable experience and I would also like to thank all my students for their interest, questions and attention. Would you let me repeat it?

Let me also thank the fantastic administrative team of the *Omega* building. They are not only doing our work easier but they also support us from the academic issues to the personal aspects.

I would also want to show my gratitud to my two English teachers. They have read and corrected my dissertation and without their help the writing would not have been of the same quality.

Similarly, I thank all the reviewers that, anonymously, have analysed and corrected my research work. Their reviews have always been constructive and, surely, I become a better researcher with their help.

I am very fortunate to have shared my doctoral period with all my colleagues and friends of the *Omega* building. They have enriched me and I have been very pleased to listen to their academic and, especially, personal experiences. The offices *S106* and *S108* have had many inhabitants but I am especially linked to Eva, Carles, Albert, Isaac, Javier, Alberto, Àlex, Eve, Jorge, Dani and Pedro. Thank you gals and guys for being my friends, for our *girl's talks* and for our amusing and funny meals!

To my dear parents, I give them my deepest gratitude and love. You always want the best for me and always give me your unconditional love and help. I do my best, but I do not know if I have been able to learn all the lessons. Please, trust me, be patient with me and be by my side as you always are while I learn from my mistakes and successes. I am very proud of being your daugther.

I express my special thanks to Jesús, who constantly tries to help me and who listened to my random thoughts when I got stuck in my research. You have been my invaluavle *rubber duck*. I deeply admire you for you erudition and for the simplicity and humility with which you explain me the concepts I do not understand. Jesús, it is almost done! Finally, I will reach the goal and become a PhD as you cheered me in your thesis.

To my dearest friends, M.Àngels, Marina and Bernat, they are my source of amusement. M.Àngels, thank you for being there, for your interest and for your *"tira, tira, tira!!"*. Marina and Bernat, I have been very fortunate to be your flatmate and your *fake daughter*, although I think that Núria is much more cuter! Marina, we will become PhD almost simultaneously. I wish you luck in your

defence!

I show my most sincere gratitude to all the mentors that have helped me from behind the scenes. In particular, I want to acknowledge the sweetness and unconditional support and love from M.Àngels and Sra. Carme. Thank you very much for your priceless help and for your lovely phrases *"Tu sempre faràs el que voldràs"* and *"Ai Maria dels Àngels, quin gust sentir la teva veu!"*.

Finally, I want to express my affection to my professors and friends from *l'Escola Politècnica Superior* at *Universitat de Lleida*. You have always been attentive to my progress. Particularly, I would like to thank all the members of the Cryptography & Graphs Research Group for giving me the opportunity to join the group. My gratitude goes beyond the academic reasons. I specially thank Prof. Miret to trust me and also for his valuable advice. And last but not least, I want to thank my dear friends Núria, Javi, Ricard, Santi and Víctor. Working with you is pleasing and enriching in all senses and not only working, but also enjoying your personal experiences, knowledge and free time. Víctor, Núria and, later, Javi and Ricard, come on, you are next!

I do not want to finish the acknowledgments without making a special tribute to Prof. Josep Maria Ribó and Prof. Joan Gimbert. You were two fantastic and modest professors that were always attentive and available to your students. It was a real pleasure to be in your lessons and learn from you.

This work has been funded by the *Universitat Politècnica de Catalunya FI 2009* grant, the *Comissionat per a Universitats i Recerca del Departament d'Innovació, Universitats i Empresa de la Generalitat de Catalunya i Fons Social Europeu FI 2009* grant and by the projects TIN2007-67982-C02-01, TIN2010-20590-C02-01, TIN2013-47137-C2-1-P and TIN2014-52211-C2-1-R of the *Ministerio de Economía y Competitividad, Gobierno de España* and *fondos FEDER*.

"Reserve your right to think, for even to think wrongly is better than not to think at all."
Hypatia of Alexandria

Pel papa i la mama;
i per tots aquells que m'han ajudat des del silenci.

Contents

Chapter 1

Introduction

"If I were again beginning my studies, I would follow the advice of Plato and start with mathematics."
Galileo Galilei

Chapter 1 Contents

Computer Graphics algorithms are present in our daily lives in many different, and even unnoticed, ways. They are a key element in the entertainment industry (Figure 1.1) and play an important role in multiple engineering fields (Figure 1.2). However, they are also used to preserve our cultural heritage (Figure 1.3) as well as a visual tool to support learning. In this sense, computer graphics can be used in medicine to visualise medical explorations of patients (Figure 1.4) or to train surgeons.

Figure 1.1: Scene from the movie *Exodus: Gods and Kings* (2014). The episode when the sea splits into two parts was digitally simulated.

The wide presence of computer graphics in our lives involves an incredible amount of versatile and powerful tools to produce, modify and show all this graphical information. From basic libraries for working with the GPU, such as OpenGL or CUDA, to complex frameworks and engines that produce graphical products. Some examples can be Photoshop, Blender, Unreal Engine or RenderMan. Facing the huge demand of graphical elements also involves multiple areas of knowledge. There are, therefore, specialists in geometry processing, in rendering, in animations and deformations, in illumination, in modeling, in fluids, etc.

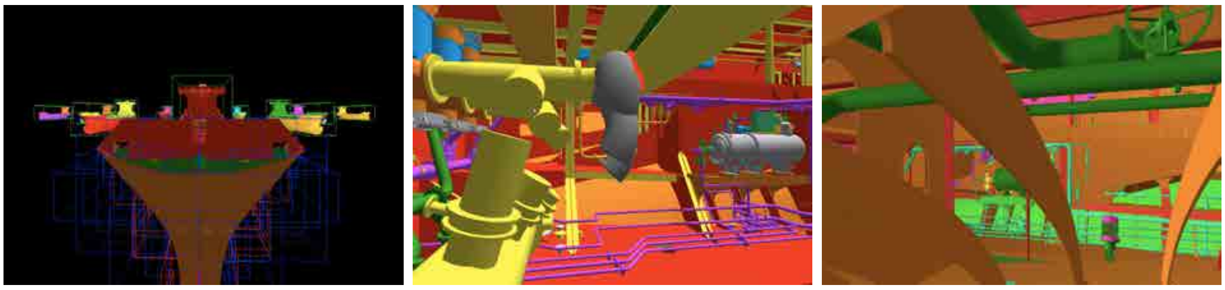


Figure 1.2: External and internal view of the hull of a ship. This sequence of images has been obtained from the Project [VNB10].

Out of the great potential and the multiple disciplines in the computer graphics field, we are particularly interested in those methodologies devoted to the modification and deformation of the 3D models. These techniques can be used to model a digital object, to animate it or to simulate some special behaviour, e.g. the heartbeat on a 3D heart model. More precisely, we are concerned with the application of such deformations over medical models, so that we can simulate the behaviour of real organs and tissues subjected to medical manipulation. These simulations can help doctors and medical specialists to understand the symptoms of their patients. Moreover, they also can be used to train these doctors before they have to deal with real situations.

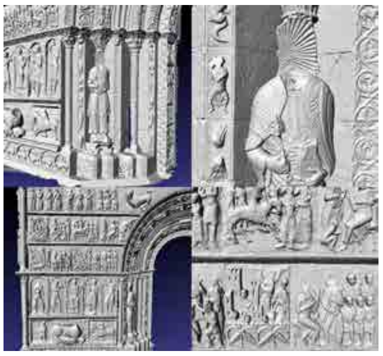


Figure 1.3: Virtual reconstruction of the *Pòrtic de Santa Maria de Ripoll*, from [CCD$^+$11].

Real organs and tissues are often soft and lack an internal rigid structure. They are elastic, so they can recover their original shape easily after being pressed or stretched, if not affected or altered by a disease that makes them stiff and inflexible. Due to their nature, this kind of biological objects are highly suitable to be simulated through 3D model spatial deformations. For this reason, we have looked further into these manipulations, especially the cage-based paradigm and the associated Generalised Barycentric Coordinates (GBC). Apart from that, biological healthy organs and tissues are incompressible. This fact means that any non-invasive medical simulated handling over a 3D representation of a real body must keep its volume constant.

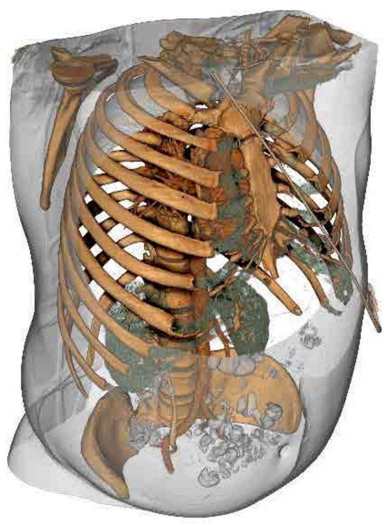


Figure 1.4: Exploration of a human trunk 3D model. Obtained from the PhD Thesis of Monclús [Mon14].

Accordingly, the present PhD Dissertation is centered in two main topics. On the one hand, it is focused on the study and the definition of new GBC systems to achieve a better reproduction of the movements and responses of the medical models. These GBC systems can be used in a cage-based paradigm but also in a cage-free spatial deformation environment, which is meant to be more flexible and easy to use from the point of view of the user, who could be a doctor, an artist, a technician, etc. The cage-free paradigm is less artificial, according to medical doctors who have participated in the present project. On the other hand, this thesis is also devoted to the study of a method to produce volume-preserving spatial deformations.

In the following sections we briefly describe the work done in the present Thesis. Section 1.1 states our motivations and the goals of our research project. Then, Section 1.2 summarises our contributions and, finally, Section 1.3 details the structure of the Dissertation.

1.1 Motivation and Objectives

According to the lines above, we want to design a new Generalised Barycentric Coordinate system oriented to the spatial deformation of biological and medical models. Furthermore, we also want to couple it to a volume-preserving algorithm that ensures the incompressibility of the models. Hence, our goal is to construct a deformation framework that meets the good properties and features below:

- **Locality:** spatial deformations often suffer from global effects, that is, the alteration that the user, e.g. an artist, induces over a certain region of the model also affects further areas. In contrast, our aspiration is to restrict, as much as possible, the modification to the regions close to the driving control points.

- **Cage-free approach:** new cage-free techniques have recently appeared to produce spatial deformation in an easier and more comfortable way. This scheme gives more freedom to the user, in so far as his or her deformation paths do not have to depend on the underlying structure of a cage any longer. In addition, the doctors asked also believe this paradigm to be more intuitive.

- **Robustness:** the system must give the same response under the same input information. Hence, the mathematical computation must be robust and stable.

- **Real-time:** the complete system is required to give real-time response during the interaction with the user. Therefore, our algorithms must be efficient.

- **Plausibility:** we want the obtained results, e.i. the resultant deformation, to be physically plausible and aesthetically pleasant. We are not looking for physically-based, correct and real output but for credible results to simulate deformations of elastic and incompressible bodies.

1.2 Contributions

As previously stated, this Thesis focuses its research on two main objectives: the definition of new Generalised Barycentric Coordinate systems that can be used in a cage-free spatial deformation environment to deform medical models and the search for a volume-preserving spatial deformation algorithm. Our contributions, therefore, can also be divided into these two areas.

Generalised Barycentric Coordinates and Cage-Free Approaches

This topic, which is studied in depth in Chapters 3 and 4, has led us to define and carefully examine some new Generalised Barycentric Coordinate (GBC) systems. All these new GBC systems are intended to be used in a cage-free spatial deformation environment, therefore, multiple approaches for this paradigm have been presented, analysed and discussed extensively. However, only those that allow us to build a deformation framework that meets the properties listed before have been collected in two different contributions

Cage-Free Spatial Deformations
M.Àngels Cerveró, Àlvar Vinacua and Pere Brunet.
VISIGRAPP - International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, 2013, Barcelona, Spain.
GRAPP/IVAPP, pages 111–114
Publisher SciTePress

3D Model Deformations With Arbitrary Control Points
M.Àngels Cerveró, Àlvar Vinacua and Pere Brunet.
Submitted

Volume-Preserving Spatial Deformations

This subject, detailed in Chapter 5, has resulted in the definition of a flexible method to produce volume-preserving spatial deformations. Its application is suitable for almost all the Generalised Barycentric Coordinate existing schemes, as well as for all the new ones proposed in this Dissertation. The obtained results were published in

Volume-Preserving Deformation Using Generalised Barycentric Coordinates
M.Àngels Cerveró, Àlvar Vinacua and Pere Brunet.
XX Congreso Español de Informática Gráfica, 2010, Valencia, Spain, CEIG, pages 57–66

1.3 Thesis Outline

The present PhD Thesis is structured in five chapters, apart from the current opening one.

- **Chapter 2: Background on Deformations via Control Points**, reviews the existing literature regarding 3D model deformations. We expose three basic paradigms to model and animate 3D objects that, nowadays, are widely used. They are the *Skeleton-Based Deformation Techniques*, the *Surface Deformation Techniques* and the *Space Deformation Techniques*. Although we present and discuss some of the main works in each paradigm, the primary focus of this chapter lies on the space deformation methods. We contextualise the cage-based paradigm and describe some of the currently most important systems of Generalised Barycentric Coordinates. Furthermore, we also consider some works in the field of *Volume-Preserving Deformations*.
- **Chapter 3: Attempts at Constructing Cage-Free Approaches**, goes through the methods we have investigated within the field of the cage-free deformations paradigm. In this chapter, we compile most of our attempts at constructing an appropriate system of Generalised Barycentric Coordinates to deform a 3D model by means of a set of handles located in its vicinity. We also analyse and discuss the mathematical reasoning behind each approach and give the strengths and weaknesses of each method, depending on the obtained results. Part of this chapter composes the publication [CVB13].
- **Chapter 4: Point-Based Celestial Coordinates**, details our main contribution. The *C-Celestial Coordinates*, the *T-Celestial Coordinates* and the *S-Celestial Coordinates* are three Generalised Barycentric Coordinate systems suitable to deform 3D models and medical models, in particular, via a set of control points placed around them. In this chapter, we mathematically derive these three systems and discuss the achieved results. These Generalised Barycentric Coordinate systems allows us to develop a deformation framework that fulfils the good properties and features listed previously. This part of our research has been submitted as a paper which is now under review.
- **Chapter 5: Volume-Preserving Deformations**, exposes our proposal to perform volume-preserving spatial deformations. We present all the mathematical reasoning to constrain the spatial deformations, so they can keep the volume of the modified model constant. This led us to define a flexible volume-preserving algorithm, which can be applied together with almost any of the existing Generalised Barycentric Coordinate systems, besides those presented in this Dissertation. This work has been collected in the publication [CVB10].
- Finally, **Chapter 6: Conclusions**, closes this Dissertation with a general review of the work done during this PhD Thesis and highlights the achieved goals. We also propose some future work to extend the presented research.

Additionally, there are two Appendices devoted to detail the long mathematical derivations related to Chapters 3, 4 and 5, respectively.

Chapter 2

Background on Deformations via Control Points

"We keep moving forward, opening new doors and doing new things, because we're curious and curiosity keeps leading us down new paths."

Walt Disney

Chapter 2 Contents

The deformation of digital models is an important topic within the Computer Graphics field. All the works in this area share the same goal, that is, to manipulate a 3D model, or a 2D shape, and deform its volume and surface until a desired pose or target is reached. These modifications are usually achieved through the use of control points or handles.

We can classify the existing methods into three big families: the Skeleton-Based Deformation Techniques, the Surface Deformation Techniques and the Space Deformation Techniques. The skeleton-based deformation techniques use bones as handles. A bone is an edge connecting two joints or vertices (Figure 2.1(b)) and it is related to the vertices of its corresponding model through a set of weights. These weights are, normally, user-defined and they determine how the model will be modified every time that the user moves a bone, i.e., how much the bone transformation affects each original vertex from the model. The surface deformation techniques use the own vertices of the 3D model, or 2D shape, as the handles of the deformation (Figure 2.3(b)). They use displacement functions to guide the transformation when one of the vertices is dragged. Finally, the space deformation techniques, which are our main focus of interest, use a set of control points that, usually, define a cage (Figure 2.6). A cage is a coarse polyhedron, or polygon in the 2D case, that encloses the model to deform. Then, the control points are related to the vertices of the model through a set of weights or coordinates. These coordinates specify the deformation induced to the model every time that a control point is displaced.

Additionally, some applications may also need to perform deformations under certain constraints. For example, they may need to apply external forces, to restrict the movements of the deformed object, to preserve some geometric properties, etc. In the case of the present Dissertation, we are interested in the study of those deformations that preserve the volume of the modified object.

The following sections present the main bibliography related to the deformation of 3D models. Sections from 2.1 to 2.3 describe the major results about skeleton-based deformations, surface deformations and space deformations, respectively. Finally, Section 2.4 is devoted to analysing the research in the volume-preserving deformations domain.

2.1 Skeleton-Based Deformation Techniques

The skin-skeleton paradigm is widely used in animation, especially, in the film and gaming industries. This paradigm animates an articulated object and its popularity relies on its easy and intuitive manipulation. According to the authors in [WSLG07], any animation process should meet three basic properties:

1. the deformation induced over the shape, or skin, must preserve local details existing in the original shape (e.g. wrinkles);
2. the deformation must be realistic, that is, it must reproduce the characteristic movements of the represented object (e.g. the contraction and relaxation of a muscle);
3. the deformation must be continuous and smooth, so the animation of the shape reproduces visually plausible and pleasant movements.

We can distinguish two different types of skeleton [Blu67, BL99, DQ04]: the *geometric skeleton* (Figure 2.1(a)) and the *inverse-kinematic skeleton* (Figure 2.1(b)). In the 3D case, the geometric skeleton, also known as *medial surface* or *medial axis*, is a 2D surface completely surrounded by the corresponding 3D object. Every point on it is a center of a sphere fully inscribed within the model and touching it at two or more distinct points. In contrast, the inverse-kinematic skeleton is composed of rigid segments, the bones, connected by joints.

There are several works focused on obtaining the skeleton of a given model in an automatic way, a process known as *skeletonization*. For example, [Blu67] and [BL99] directly compute the medial axis of the object; [LV99] extracts the skeleton by means of level set functions; [ZT99] uses a voxelization; [KT03] and [LKA06] firstly compute a shape or mesh decomposition and, then, get the skeleton that connects the neighbouring components and [WML$^+$03] computes the skeleton with the aid of a simulation of repulsive forces.

Once the skeleton is obtained, it can be used to deform or animate its associated model. The most used technique of this family of deformation algorithms is the Skeletal Subspace Deformation (SSD). The SSD method normally uses the inverse-kinematic skeleton to perform the deformations. It links each of the vertices, or points, $\mathbf{p}$ of the skin surface to the skeletal joints through a set of weights, which, usually, are defined and tweaked by the user, e.g. an animator or artist. When the skeleton is manually modified to a pose, the points $\mathbf{p}$ are deformed by means of a linear combination using these weights. Although SSD can achieve realistic deformations, as muscle bulging, by manually adjusting the weights until the resulting deformation is satisfactory, the constant adaptation of

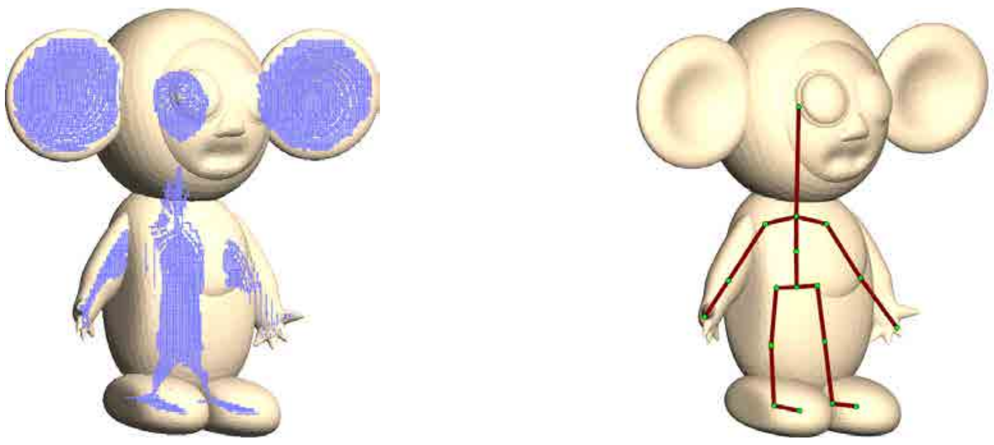

(a) Geometric Skeleton (b) Inverse-Kinematic Skeleton

Figure 2.1: Examples, from [BP07], of a geometric skeleton in (a) and an inverse-kinematic skeleton in (b).

the weights makes the method tedious and even frustrating. In addition, the linear nature of the blending of the points **p** causes the unpleasant effect of collapsing joints, also known as the candy-wrapper artifact (see Figure 2.2). A complete overview of the SSD algorithm can be found in [MTLT88], where the authors use it to animate hands. Other works like [BL99] and [YBS07] use the combination of both defined types of skeletons to animate characters. Specifically, [YBS07] uses the SSD technique to deform the geometric skeleton by means of the inverse-kinematic skeleton. Then, the deformation of the geometric skeleton is transferred to the skin surface, the 3D object, using discrete differential coordinates.

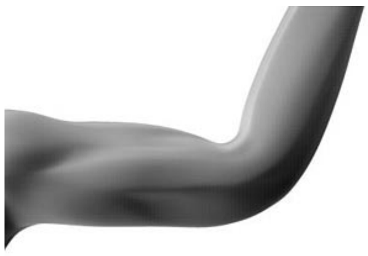


Figure 2.2: Example of the candy-wrap artifact. This figure belongs to [LCF00].

New research in this deformation paradigm has focused on overcoming the drawbacks of the SSD [YBS03, YHM06, KCvO08, JS11]. In particular, they try to avoid the laborious task of manually adjusting the weights that link the skeleton with the model. For example, [YBS03] combines a geometric skeleton with Free-Form Deformation (FFD)[1] techniques to achieve the desired results. Firstly, the method deforms the skeleton using one of the existing FFD methods. Then, the 3D model is fitted to the modified skeleton through a set of displacements. These displacements have been previously computed from the rest pose of the model and its skeleton. On the other hand, in [YHM06], the authors link an inverse-kinematic skeleton directly to the simplexes of the 3D model, instead of its vertices. Every time that the skeleton is deformed, they find the transformation relating the initial and final position of each bone and apply it to the corresponding

[1]A more detailed information about Free-Form Deformation techniques can be found at Section 2.3.

simplexes. Then, they proceed to an optimisation step to stitch these simplexes together while respecting the transformation as much as possible.

A step forward from the SSD is achieved by the Pose-Space Deformation (PSD) techniques. PSD combines the SSD with a set of displacements supplied by examples. These examples can be defined by an animator who designs a few key poses and manually deforms the skin until the correct shape is obtained. However, they also can be obtained by a 3D acquisition step followed by the corresponding rigging process to relate the real movement with the underlying skeleton. Research on PSD techniques can be found in [LCF00, SRC01, RLN06, WSLG07, BLB$^+$08].

Finally, other authors [BP07, HRE$^+$08, JZvdP$^+$08, OZS08, BTST12] have also approached the problem from another point of view and they have focused their research on finding ways to reuse a given inverse-kinematics skeleton with an associated set of motions and poses. Their main goal is to automatically adapt a generic skeleton to a character and link it to the model surface.

2.2 Surface Deformation Techniques

The surface deformation paradigm typically treats a 3D model as a two-dimensional surface S, which describes the boundary of the object and is usually represented as a triangular mesh or a point cloud. The technique permits the user, e.g. an artist, to directly edit this surface by means of its own vertices. That is, the handles or control points of the deformation are the vertices, or points, of the model[2], as shown in Figure 2.3. When the user picks and drags one of the vertices of the model, the whole surface is modified according to a displacement function. This displacement function defines how much a point, or vertex, on the surface is transformed depending on the new location of the last moved handle. The induced deformation is continuous, smooth and local. Moreover, many of the techniques in this paradigm also seek shape and detail preservation with respect to the original surface of the model. The local maintenance of these properties produces aesthetically pleasing and even physically plausible deformations.

The bibliography in the surface deformation research field proposes different approaches to build displacement functions. One of them is by means of differential coordinates, which define any point, or vertex, $\mathbf{p}$ of the surface S with respect to its neighbours. They can be seen as a measure of how different a property in $\mathbf{p}$ is with respect to the same property in its neighbouring points. Thus, when a user moves a handle, all the points on S adjust their new locations in such a way that they keep their new differential coordinates as similar as possible to the initial ones. Differential coordinates can be computed in several ways, such as the use of the Laplacian operator shown in [SCOL$^+$04, LSCO$^+$04, NSACO05, Sor06, ZHM11] or the gradient in [YZX$^+$04, ZRKS05].

Particularly, the authors in [SCOL$^+$04] deform the surface of an object, transfer and mix geometric details between two different surfaces and even transplant a part of a surface onto another one through the use of *Laplacian coordinates*. The Laplacian coordinates represent the difference between the position of a given point $\mathbf{p}$ and the average position of its neighbours. They are computed with the Laplacian operator of the triangular mesh that defines the surface. The authors also apply an appropriate transformation to the vertices in order to make the Laplacian invariant to locally linearised rigid transformations and scaling. Then, whenever the user moves a handle, they solve an error minimisation problem between the original Laplacian coordinates and the current ones by

[2]Most recent techniques also permit other kind of handles as curves, which can be extracted from the surface or painted on it.

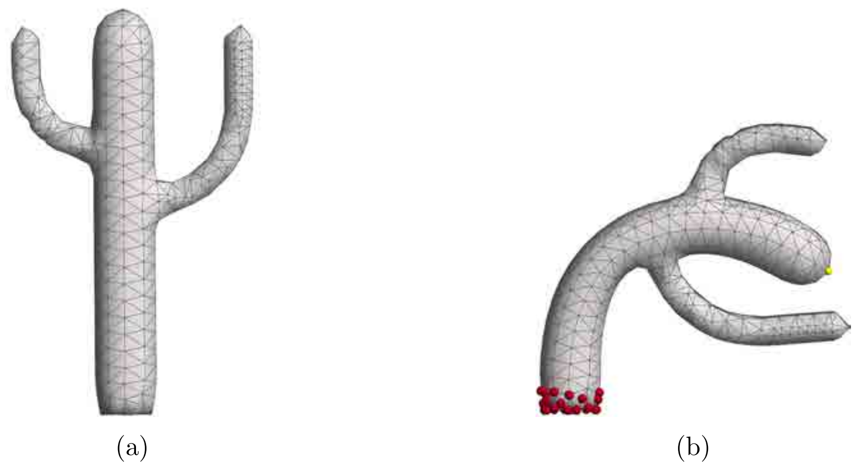


Figure 2.3: Example of handles used in a surface deformation process, from [SA07]. (a) shows the original model of a cactus, whose surface is represented as a triangular mesh. (b) presents an example of deformation of the cactus. While the red handles are constrained to keep in place, the yellow one at the top is translated to yield the deformation.

means of least squares method. This minimisation process leads to the new positions of the vertices of the model.

The pipeline in [YZX$^+$04] is very similar to that in [SCOL$^+$04]. However, in this case the authors use the Poisson equation and the gradient field of a parameterised mesh instead of the Laplacian operator. They can also deform a single surface or merge a partial surface onto another.

Another way to build displacement functions is through an optimisation problem over a defined energy [KCVS98, BK03, BK04, BPGK06]. Usually, the studied energy function corresponds to an elastic energy that quantifies the stretching and the bending capabilities of the surface. Some research works within this context, e.g. [KCVS98] and [BK03], propose a multiresolution editing paradigm which consists in decomposing the original 3D model into two parts: a low resolution surface and a set of displacements that store the geometric detail of the initial model. Specifically in [BK03], the authors simplify the input surface S, which is represented by a triangular mesh, to a coarse surface S_c and store the details using displacement volumes. A displacement volume is a prism that relates each triangle $T = \{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k\}$ of the original surface S with the points $\{\mathbf{p}_i, \mathbf{p}_j, \mathbf{p}_k\}$ on the coarse surface S_c such that $(\mathbf{v}_i - \mathbf{p}_i)$ is normal to S_c. Then, they deform S_c by means of the constrained energy minimisation, as in [KCVS98]. Once the coarse surface S_c is deformed, the details are restored iteratively, moving the original points $\mathbf{v}_i$ to a new position $\mathbf{v}'_i$ that preserves the volume of the stored prisms. Other examples of the multiresolution editing paradigm can be found at [KBS00], [BSPG06] and [MBK07]. Figure 2.4 shows an example of this scheme.

Furthermore, PriMo [BPGK06] also uses an elastic energy to deform the surface of the model. However, this technique does not modify this surface directly but, instead, deforms a set of rigid prisms that wrap it. Assuming the surface is represented as a triangular mesh, the prisms enclosing it are obtained by an extrusion along its vertex normals, which results in a prism per mesh face. Then, they are coupled through an elastic energy function. Hence, when the user moves or orientates a subset of prisms, the system solves the optimisation problem that keeps them rigid and minimises the energy. They are kept rigid to prevent their geometry from degeneration and ensure numerical robustness of the method. Finally, the new surface is derived from the resulting prisms by a process

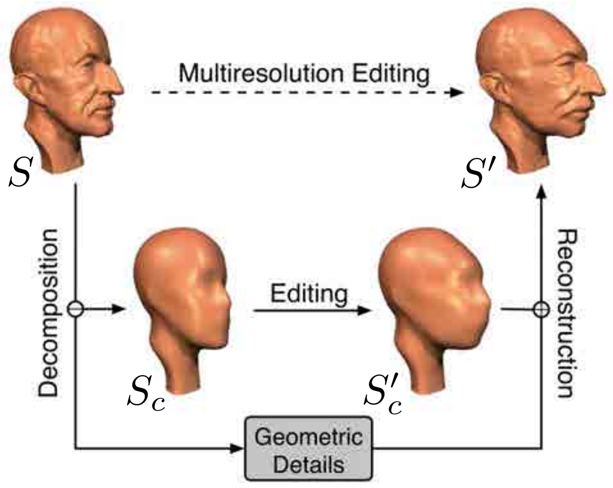


Figure 2.4: Multiresolution editing pipeline extracted from [MBK07].

that updates the positions of its vertices with the average transformation of its incident prisms.

Beyond the presented methods, there also exist different schemes, comprised in the surface deformation paradigm, that belong to the *as-rigid-as-possible* (ARAP) techniques [ACOL00, IMH05, SA07, SDC09, ZHM11, PLHY13, LG14]. As-rigid-as-possible methods are also based on the minimisation of an elastic energy, which tries to keep the rigidity of the local transformations during the deformation process. This rigidity makes it possible to preserve the surface details.

The ARAP method presented in [SA07] divides the surface of the model, represented as a triangular mesh, into overlapping cells. A cell consists of all the triangles incident to a particular vertex of the mesh, that is, a cell is equivalent to the first ring of that vertex. According to this, the authors consider a transformation to be rigid if, given an original cell C_i and its corresponding deformed cell C'_i, there exists a rotation matrix $\mathbf{R}_i$ that transforms C_i to C'_i. Thus, the deformation framework consists in finding optimal rotation matrices $\mathbf{R}_i$ such that they minimise the error between the original and the transformed cells C_i and C'_i, for all the cells that cover the surface of the model.

All the methods presented until now belong to the group of *discrete deformations*, a term introduced by Martinez-Esturo et al. in [MERT12]. Discrete deformations only compute the final position of the vertices and do not consider the deformation path leading from the initial state to the final one. In contrast, *continuous deformations* integrate vertex positions along smooth vector fields. Although most of the bibliography concerning surface deformation methods describes techniques that belong to the discrete deformations paradigm, there exist some research works that also expose how to compute vector fields regarding some constraints, e.g., volume and shape preserving, and how to use them in continuous deformation processes. Some examples are [vFTS06, vFTS07, KMP07, MERT12].

Finally, shape and surface analysis can also be used in other applications beyond deformations. For example, creation of texture atlases [ZSGS04], mesh segmentation [LZ07, dGGV08], shape correspondence [JZ07], mesh compression [VMHB14] and sketch based model edition [NSACO05, NISA07]. Moreover, surface deformation techniques are also used as powerful tools in other research works [BWKS11, BBB$^+$14]. In [BWKS11] a structure-aware shape deformation is presented. This technique allows for detection of regular patterns in the initial model and preserve them during the deformation process, minimising the distortion. Once the patterns are identified, the method computes an elastic deformation based on [SCOL+04] and [SA07]. On the other hand, the authors in [BBB+14] present a technique for adding fine-scale details and expressiveness to low-resolution facial models. Firstly, they capture a representative performance of an individual. Then, they extract an *expressiveness model* that encodes the subtle spatial deformation details that occur during that capture process. Finally, these deformations are transferred to low-resolution models by means of a matching process and a surface reconstruction step based on [BSPG06].

Although the results obtained through surface deformation techniques are intuitive and high quality, they have a major drawback: their high computational cost. This disadvantage can make them useless for real-time environments, where interactivity is a key requirement.

2.3 Space Deformation Techniques

The space deformation paradigm, unlike surface deformations that directly manipulate the embedded object, operates on the ambient space enclosed by some control structure. Commonly, this control structure takes the form of either a lattice or a coarse polyhedron, or polygon in 2D, which resembles the shape of the wrapped model and is called *cage*. The vertices of the lattice or the cage are the handles, or control points, used to deform the enclosed space. Figure 2.5 shows an example of a lattice while Figure 2.6 shows an example of a cage.

This deformation scheme was first introduced by Sederberg and Parry [SP86] as the Free-Form Deformation (FFD) paradigm. The FFD method was designed to sculpt and model solid objects defined by an analytic surface. However, it can also deal with polygonal models. It encloses the object into a lattice, as shown in Figure 2.5, and relates it with respect to the control structure through a local system of coordinates. Hence, when the user moves any of the vertices $\mathbf{v}_i$ of the lattice, which are the control points, any point $\mathbf{p}$ inside it, particularly those on the surface of the object, is modified according to the trivariate tensor product Bernstein polynomial.

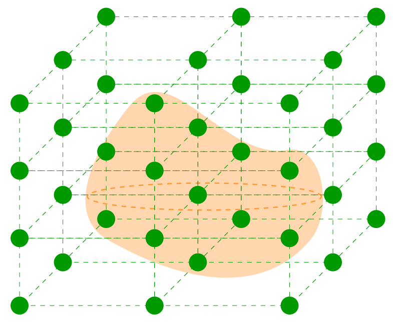
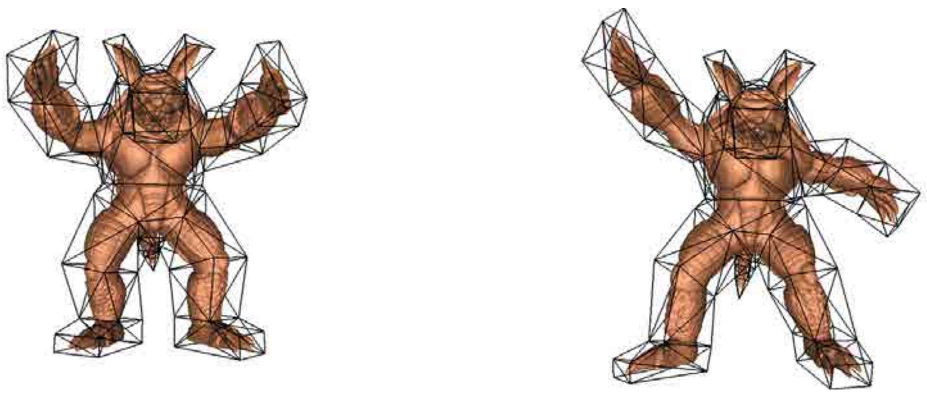


Figure 2.5: Example of a Free-Form Deformation lattice surrounding an object. Its vertices are the handles used to drive the deformation.

This seminal work was extended by Coquillart [Coq90], MacCrackent et al. [MJ96] and Kobayashi et al. [KO03] who, gradually, make the underlying lattice structure more flexible until achieving a triangular mesh with arbitrary topology in [KO03]. However, the FFD method and its rigid control structures make the deformations not as flexible as desired. Thus, the research in this field focused on the study of more general control structures that fit the wrapped model better and with more degrees of freedom to perform the deformations in an easier and more intuitive way.

From this search for new methods arises the *cage-based* deformation methods which, as stated above, surround the model into a coarse polyhedron called cage. Figure 2.6 shows an example of a cage used to animate the model of the *Armadillo*.



(a) (b)

Figure 2.6: Example of a cage, whose vertices are the handles, used in a space deformation process, from [JSW05]. (a) shows the original model of the *Armadillo*, enclosed inside the cage, which is represented as a triangular mesh. (b) presents an example of deformation of the *Armadillo*.

Cage-based deformation techniques construct space deformations by means of Generalised Barycentric Coordinates (GBC). Barycentric coordinates are a very useful mathematical tool and allow us to interpolate continuous data over a space domain from discrete or continuous values defined on the boundary[3] of this domain. Thus, assuming that the model is represented by a triangular mesh for the sake of simplicity, its vertices $\mathbf{p}$ are defined as a convex combination of the vertices $\mathbf{v}_i$ of the cage through the use of GBC

$$\mathbf{p} = \sum_i \mathbf{v}_i \alpha_i(\mathbf{p}), \tag{2.1}$$

where $\alpha_i(\mathbf{p})$ is the GBC of $\mathbf{p}$ with respect to the i-th control point $\mathbf{v}_i$. Usually, these coordinates are required to fulfil a set of basic properties:

- Reproduction of the identity: the point $\mathbf{p}$ is defined through the linear combination shown in Equation (2.1).
- Reproduction of the unity: the coordinates $\alpha_i(\mathbf{p})$ must sum 1, that is $\sum_i \alpha_i(\mathbf{p}) = 1$.
- Positivity: the coordinates $\alpha_i(\mathbf{p})$ must be positive, so $\alpha_i(\mathbf{p}) \geq 0 \ \forall \alpha_i(\mathbf{p})$.
- Continuity and smoothness: the functions to compute the coordinates $\alpha_i(\mathbf{p})$ must be continuous and smooth. We require, at least, C^1 continuity.
- Locality: the influence of the control points over the points inside the cage must be local. That is, the control points should have a larger impact on the points near them than on those that are located farther away.

Moreover, in contrast to the surface deformation paradigm, the GBC only need to be computed once, so the cage-based deformation pipeline consists of two steps, as shown in Figure 2.7:

[3]In the deformations context, this boundary is exactly the surface of the cage.

1. Deformation set-up: the coordinates are computed and stored for their further use. This step is also known as the *binding process* because it establishes the relation between the model and its cage.

2. Deformation process: every time that the user drags a control point and modifies the cage, the mesh inside it is recomputed using the new location of the cage vertices and the GBC previously computed.

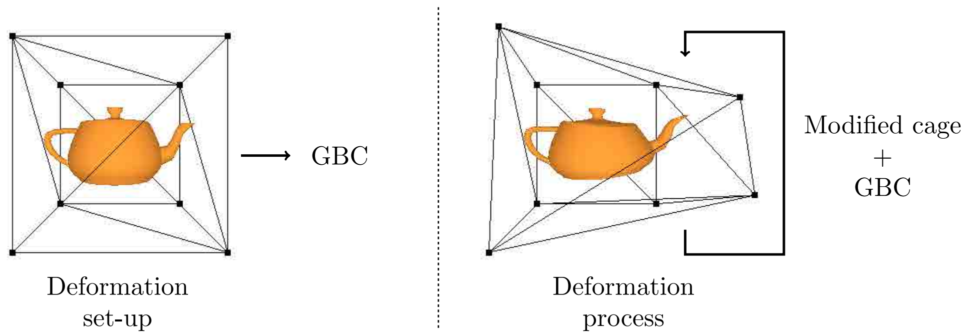


Figure 2.7: Pipeline of a cage-base deformation process.

This pipeline makes the method highly efficient and, therefore, cage-based deformations can achieve real-time performance.

Due to their importance in the space deformations field, there is an extensive bibliography proposing different GBC schemes. The classical Barycentric Coordinates were first introduced in 1827 by Möbius [Mö27]. They define a point $\mathbf{p}$ inside a simplex[4] with respect to its vertices. However, the restriction over the polytope structure has led to numerous extensions of this scheme in the case of more general cages. Wachspress [Wac75], Meyer et al. [MBLD02], Ju et al. [JSWD05] and Floater et al. [FHK06] defined GBC systems that work over convex polygons and polyhedra. Moreover, [WSHD07,SJW07,MS10] allow cages with curved boundaries and [MS10] also allows non-closed and self-intersecting polytopes. However, it was not until the definition of the Mean Value Coordinates (MVC) that a solution to compute coordinates over concave polytopes was given. Floater et al. [Flo03, HF06] first defined the MVC in the 2D case and, then, two contemporary works, Floater et al. [FKR05] and Ju et al. [JSW05], generalise them to 3D polyhedra domains. Both works derive the MVC by the observation that the integral of all unit normals $\vec{\mathbf{n}}$ around a sphere S is zero

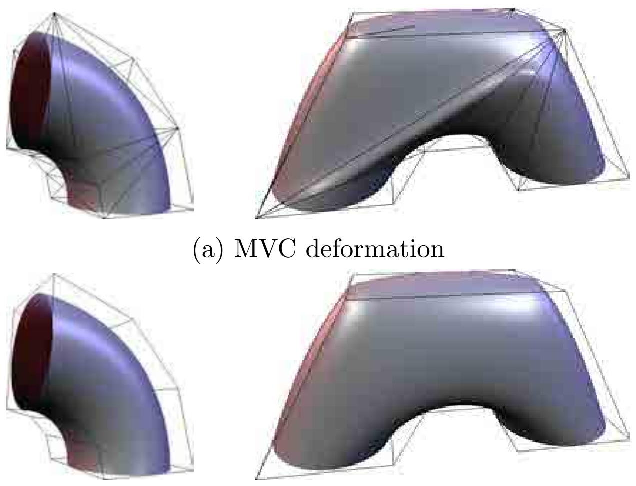
$$\int_S \vec{\mathbf{n}} = \vec{\mathbf{0}}$$

and compute this integral with the help of the classical barycentric coordinates adapted to spherical triangles. The resulting coordinates are well-defined, that is, they meet all the GBC properties listed before, and are positive in the interior of the polytope, if it is convex, or in its kernel[5], if it is concave. They are C^∞-continuous inside and outside the polytope, or its kernel, and C^0-continuous on its boundary. This characteristic can produce sharp features when the MVC are used in cage-based deformations, as shown in Figure 2.8(a). To alleviate this situation, Langer et al. [LBS06] developed the Spherical Mean Value Coordinates (SMVC) as an extension of the MVC that permits

[4]A triangle in 2D and a tetrahedron in 3D.

[5]Assuming the facets of a polytope as opaque walls, the kernel is the volume inside it from which all its vertices can be viewed directly.

non-triangulated polyhedra. That is, the SMVC can be computed over a polyhedron whose faces are arbitrary planar polygons, so they are not restricted to triangular faces (see Figure 2.8(b)).


(a) MVC deformation

(b) SMVC deformation

Figure 2.8: Example of a MVC deformation in (a) and a SMVC deformation in (b) taken from [LBS06]. Notice the artifacts introduced by the triangulation of the cage in the MVC case and how they disappear with the help of a more flexible cage and the SMVC.

As previously stated, the MVC are only positive in the interior of convex polytopes and inside the kernel of concave ones. In addition, their computation depends on the euclidean distance between the point $\mathbf{p}$ and the handle $\mathbf{v}_i$, which ignores the visibility that the points inside the polytope have over the control points. These two features cause undesired slimming and a global effect when the MVC are used to deform a 3D model enclosed in a strongly concave cage. Figure 2.10(b) shows how the ring finger tightens when only the section of the cage around the middle and index fingers is modified.

In order to solve this problem, two new systems of coordinates have been developed: the Harmonic Coordinates (HC) [JMD$^+$07] and the Positive Mean Value Coordinates (PMVC) [LKCOL07]. The HC are the unique solution to the Laplace equation with Dirichlet boundary conditions. This solution is a harmonic function, hence, the name of these coordinates. They are non-negative, more local than the MVC, C^∞-continuous inside the cage and C^0-continuous on its boundary. However, they do not posses an analytic expression, so they are typically approximated using a discretisation of the interior of the cage. In contrast, the PMVC reach the non-negativity property based on the same principles as the MVC. Nevertheless, in order to fulfil this requirement they must relax the constraints of smoothness and continuity through the supporting planes of the cage. Figure 2.9(b) shows the singularities that the PMVC present near the concavities of the cage.

Figure 2.10 shows an example of a deformation of the middle and index fingers of a 3D hand. This deformation is driven by the MVC, the PMVC and the HC respectively. Notice how the process that uses the MVC causes unintuitive effects over the ring finger. In contrast to this, the PMVC and the HC do not modify the ring finger, so the resulting deformation is more local, intuitive and aesthetically pleasant.

The coordinate systems presented above express the point $\mathbf{p}$ inside a cage as an affine combination of this cage vertices $\mathbf{v}_i$, as shown in Equation (2.1). This means that the cage produces affine transformations, which may include shear, anisotropic scale and self-intersections. In this context,

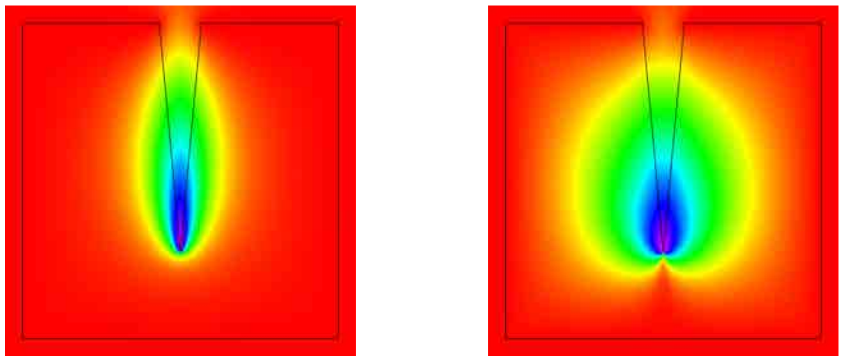


Figure 2.9: Comparison, from [LKCOL07], of the smoothness and continuity of the MVC and the PMVC. Notice the singularity of the PMVC in the vicinity of the concavity of the cage.

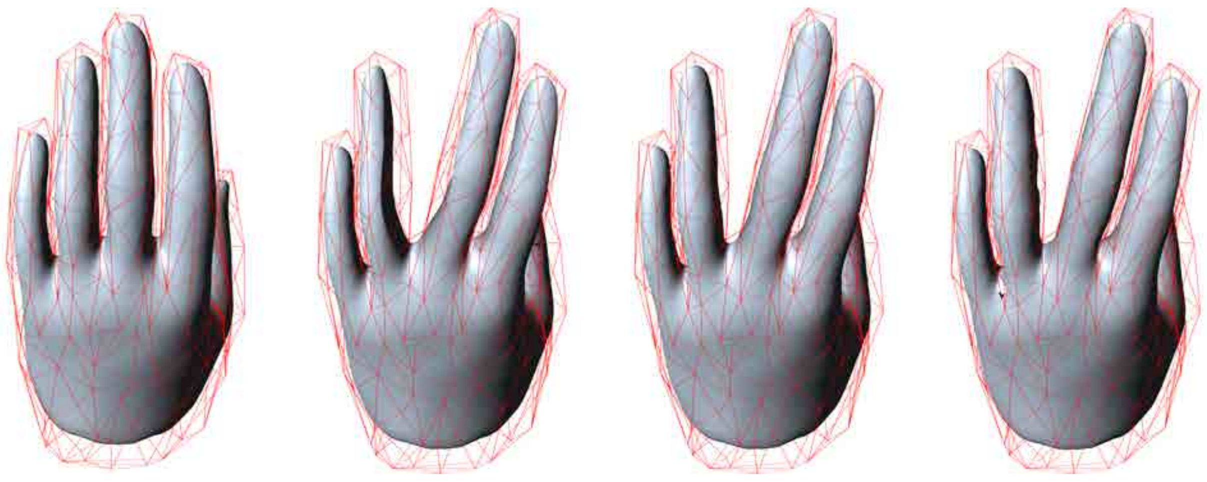


Figure 2.10: Example of a deformation process over a 3D hand. It compares the obtained results depending on the system of GBC used: the MVC in (b), the PMVC in (c) and the HC in (d). This sequence has been obtained from [LKCOL07].

intuitive control over local rotations and shears and, moreover, achieving shape and detail preservation of the deformed 3D models are unfeasible. Hence, in order to increase the control over the modification process and to perform shape-preserving space deformations, Langer et al. [LS08] and Lipman et al. [LLCO08, LL10] developed the Higher Order Barycentric Coordinates and the Green Coordinates, respectively. On one hand, Higher Order Barycentric Coordinates try to increment the control over the deformation handles by taking into account the derivatives of the coordinate functions, so they produce an Hermite interpolation at the control points. Particularly, the authors in [LS08] formulate the Higher Order Mean Value Coordinates, although their technique can be applied to other systems of coordinates like the HC. On the other hand, the Green Coordinates (GC) are devoted to achieving shape-preserving space deformations. They take into account the orientation of the faces of the cage, together with the position of its vertices, so they express the point $\mathbf{p}$ as the linear combination

$$\mathbf{p} = \sum_i \mathbf{v}_i \alpha_i(\mathbf{p}) + \sum_j \vec{\mathbf{n}}_j s_j \beta_j(\mathbf{p}),$$

where $\vec{\mathbf{n}}_j$ are the normals of the faces of the cage, s_j are scaling factors to keep scale invariance and $\alpha_i(\mathbf{p})$ and $\beta_j(\mathbf{p})$ are the GC of the point $\mathbf{p}$. The GC do not interpolate the boundary of the cage. However, this is a necessary deal so they can produce conformal mappings in 2D and quasi-conformal mappings in 3D, which leads to shape-preserving deformations. Figure 2.11 shows a comparison between the GC, the HC and the MVC. Notice how the GC produce smoother and more pleasant deformations than the previous systems.

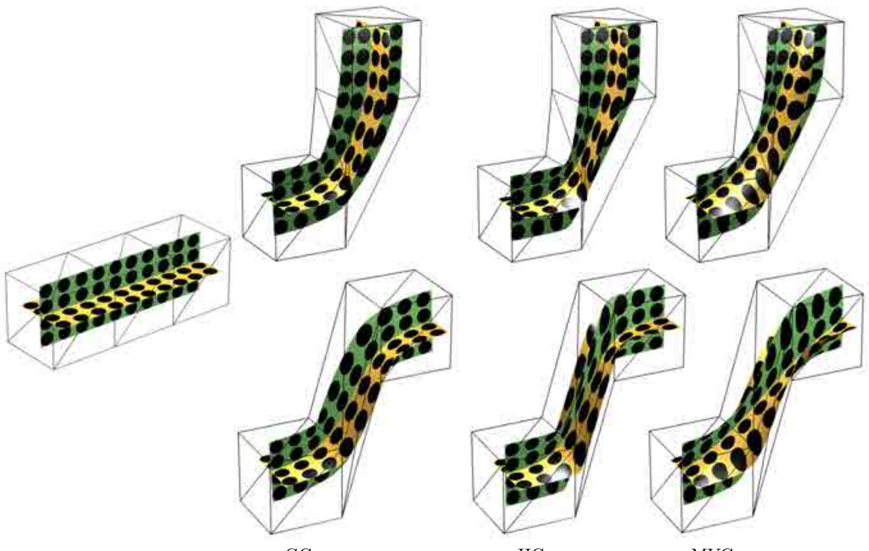


Figure 2.11: Comparison of the same deformation driven by the GC, the HC and the MVC, extracted from [LLCO08].

Later, Weber et al. [WBCG09] demonstrate that the GC belong to the Complex Barycentric Coordinates (CBC). The CBC are a family of coordinates whose members are characterised by the fact that they are barycentric coordinate systems with complex numbers.

More recently, the research in the deformations field and, particularly, in the space deformations field, has focused on making the handles of the process more flexible: developing *cage-free* methods [LLD$^+$10], combining multiple kinds of handles [JBPS11] and localising the deformation [GGPCP13, ZDL$^+$14].

In their work, Li et al. [LLD$^+$10] use the GC to deform a 3D model, which is represented as a triangular mesh. They replace the supporting cage by an umbrella shaped cell. This umbrella is completely transparent to the user and is automatically constructed and updated when he or she directly drags a vertex of the model. This method simulates the surface deformation paradigm in terms of flexibility of the handles. It also performs local shape-preserving deformations in real-time, due to the use of the GC.

In [JBPS11], the authors propose the Bounded Biharmonic Weights, that allow multiple deformation handles. The user can operate with cages, skeletons and isolated control points on the surface of the model to accomplish the desired deformation. Their method reaches its goals through a space discretisation, which results in a tetrahedralisation of the deformation domain that includes the model and all the handles, and the minimisation of a Laplacian energy.

Finally, García et al. [GGPCP13] present a multi-cage system to restrict the deformations in more localised regions. Their technique also increases the continuity of the coordinates across the faces of the connected cages by computing a blending function. Furthermore, they also allow the combination of different GBC to make the deformation even more flexible. In a similar way, Zhang et al. [ZDL+14] also localise the effect of the deformation. However, instead of connecting cages as in [GGPCP13], they have a single cage and, for each point $\mathbf{p}$ of the model, they compute its coordinates with respect to a small set of control points. That is, the points of the model are not affected by all the handles but only by a small set of them.

2.4 Volume-Preserving Deformations

In the previous sections we have seen that detail and shape-preserving deformations are an outstanding research topic. However, these are not the only attributes that are intended to be preserved. In recent years there has been a growing interest in performing deformations that conserve other kinds of properties of the models, e.g. their volume. Constrained manipulations of the objects contribute to producing more realistic and physically plausible deformations.

Zhou et al. [ZHS+05] extend the surface deformation techniques in [SCOL+04] and [YZX+04] to a volumetric approach. Given a particular model represented as a triangular mesh, they construct a volumetric graph. First, they build an interior layer by offsetting each vertex of the input mesh in the opposite direction to its normal. Then, they fill this layer with a lattice and connect all the vertices, as shown in Figure 2.12. Once the graph is obtained, the technique encodes the volumetric details as the difference between each of its points and the average of their neighbours. Every time that the user drags a deformation handle, the system solves the new positions of the vertices of the model by means of a least-squares minimisation. This process preserves surface details and avoids unnatural volume changes.

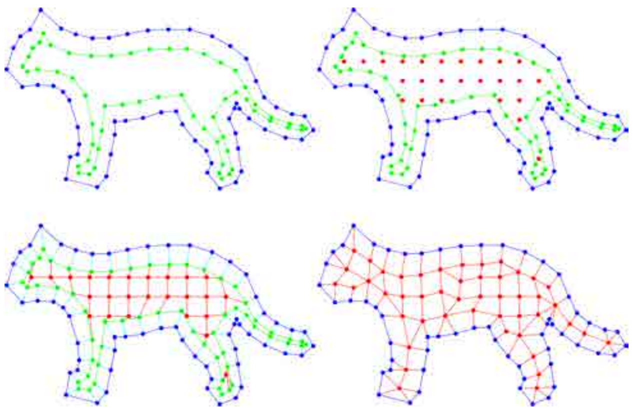


Figure 2.12: Construction of the volumetric graph, from [ZHS+05].

The methodology in [vFTS06], which has already been introduced in Section 2.2, produces surface deformations via a divergence-free vector field. Thus, the deformation preserves the volume of the modified model. Moreover, the technique in [BK03], also presented in Section 2.2, avoids extreme changes in the volume of the manipulated model due to the use of displacement volumes in its multiresolution paradigm.

Lipman et al. [LCOGL07] preserve the volume through the scale of the transformations, according to local curvature information. Figure 2.13 shows the same deformation before and after the volume correction.

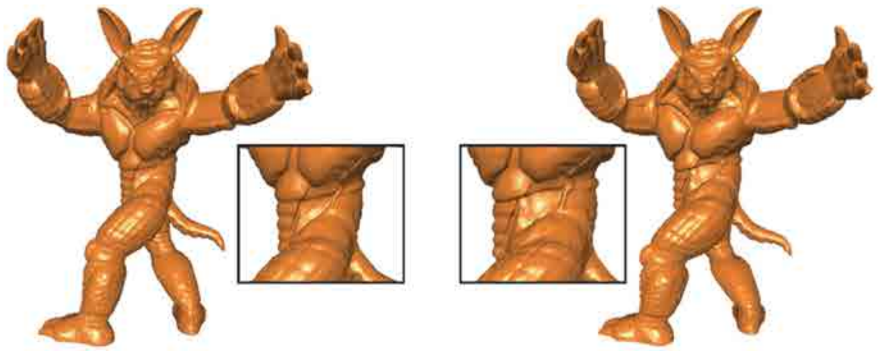


Figure 2.13: Example of a deformation of the *Armadillo* model without volume correction (left) and with volume correction (right). This image has been extracted from [LCOGL07].

Additionally, there are some space deformation techniques that also preserve the volume of the modified object. Aubert et al. [AB97] present a volume-preserving FFD technique based on the DOGME [BB91] model. DOGME deforms an object placing some handles on the desired locations and guaranteeing some constraints through an optimisation parameter. The volume is preserved with an iterative algorithm that keeps it constant while minimising the norm of this parameter. In [HML99], Hirota et al. propose an extended FFD method that uses an elastic spring energy connecting the handles of the lattice. When the user moves one of these handles, the lattice is updated according to the constrained minimisation of the associated energy, subject to null volume deviation. Once the lattice is updated, the deformation is transferred to the object surface as done in the classical FFD technique, that is, by means of the trivariate tensor product Bernstein polynomial. Other FFD techniques that preserve the volume are [RSB95] and [WM14].

Huang et al. [HSL$^+$06] present a technique that mixes a nonlinear energy minimisation with a cage-based deformation. The authors propose building a coarse control mesh, i.e. a cage, around the original mesh of the model and deform it according to the energy minimisation. This energy encodes the linear and nonlinear constraints over the deformation, particularly, it permits the exact volume preservation of the modified object. Once the cage is deformed, it transfers the changes onto the original mesh via a mean value interpolation.

In [ACWK06], Angelidis et al. define volume-preserving deformations by means of *swirling-sweepers*. A sweeper [AWC04] is a space deformation technique that simulates clay modeling. When the user moves a modeling tool across a desired path, the model is deformed along this path, according to the shape of the used tool. Then, a swirl-sweeper is a sweeper restricted to a point tool which can be rotated an angle θ around an axis $\vec{\mathbf{r}}$ and twists space locally, without compression or dilation of the deformed model.

Finally, there is also an increasing interest in producing volume-preserving skinning to achieve more realistic animations. The authors in [vFTS08] apply a volume correction step after each skinned deformation. This correction consists in moving the vertices of the deformed model, represented as a mesh, along an automatically constructed displacement field. In addition, they also allow the user to interactively control the volume correction, so some custom effects can be obtained, e.g., muscle bulging. Other works in this area are [AS07] and [RHC09].

Chapter 3

Attempts at Constructing Cage-Free Approaches

"I have not failed. I've just found 10,000 ways that won't work."
Thomas A. Edison

Chapter 3 Contents

In recent years, new deformation algorithms for 3D models have appeared with the goal of making the process more intuitive. These new methods are focused on increasing the flexibility and simplifying the deformation handles, so they can perform volume deformations without the need of a complex structure, e.g. a cage. They only need the definition of a set of control points in the vicinity of the model, which we assume represented as a polygonal mesh. Then, a set of weights can be computed for every vertex of the model. These weights, also called coordinates, define the vertices of the model as a linear interpolation of the control points, making it possible to deform the model through the direct manipulation of these handles.

In [JBPS11], the Bounded Biharmonic Weights (BBW) permit the users to mix isolated control points, skeletons and partial cages. On the other hand, the technique in [LLD$^+$10] only works with isolated control points over the surface of the model. To reach the desired deformation, this method automatically builds an open, umbrella-shaped cage around the activated control point, the one that is being manipulated by the user. The final deformation is, therefore, driven by this open cage, in a completely transparent way for the user.

Within this context, this chapter describes different attempts to construct cage-free deformation approaches. All of them try to define a system of coordinates with respect to the control points that satisfies some basic properties:

- Reproduction of the identity: given a set of control points $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ and a point $\mathbf{p}$

inside their convex hull, $\mathbf{p}$ can be defined through the linear combination

$$\mathbf{p} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i, \tag{3.1}$$

where $A = \{\alpha_1, \ldots, \alpha_n\}$ are the coordinates of $\mathbf{p}$ with respect to the control points.

- Reproduction of the unity: the weights α_i, computed for any point $\mathbf{p}$ inside the convex hull of the control points, must sum 1

$$\sum_{i=1}^{n} \alpha_i = 1, \quad \forall \mathbf{p}. \tag{3.2}$$

- Positivity: the weights α_i, computed for any point $\mathbf{p}$ inside the convex hull of the control points, must be positive

$$\alpha_i \geq 0, \quad \forall \alpha_i. \tag{3.3}$$

- Continuity and smoothness: the functions to compute the weights α_i must be continuous and smooth. We require, at least, a C^1 continuity.
- Locality: the influence of the control points over the points inside their convex hull must be local. That is, the control points should have a larger impact on the points near $\mathbf{p}$ than on those that are located farther away.

Notice that the reproduction of the identity, the reproduction of the unity and the positivity properties ensure that Equation (3.1) is, in fact, a convex combination.

Firstly, we tried to develop a system of coordinates from the well-known Mean Value Coordinates [FKR05] (MVC). We substituted the fixed, user-defined cage that supports the MVC computation for a given point $\mathbf{p}$ of the model, by an automatically computed connection between the control points. This imposed connectivity depended on the point of view of $\mathbf{p}$ so it was not constant for all the points of the model. The MVC, like most of the cage-based coordinate systems [LBS06,JMD+07, LLCO08], strongly depends on the connection between the control points so that two different connections result in two completely different sets of coordinates. Accordingly, the flexibility on the connectivity provided by our methods induced discontinuities in the functions of the coordinates when the computation of these coordinates were based on the MVC equations.

Trying to improve the Mean Value based methods, our next attempts did not compute a different connectivity per point of the model. They worked over a single, constant and layered connectivity that tried to take into account all the possible connections between the control points. However, these approaches resulted in rude coordinates that did not have the required level of smoothness to perform intuitive and visually pleasant deformations.

Finally, we decided to work without any kind of supporting connectivity between the control points. We, therefore, addressed the problem directly in the n-dimensional space, where n is the total number of existing control points.

To sum up, our approaches can be classified into three different groups: the methods based on the Mean Value Coordinates [FKR05], the layered methods and the n-dimensional methods. In the following sections we describe all the studied methodologies and discuss their advantages and disadvantages. Finally, a summary of our results and conclusions can be found in Section 3.4.

3.1 Mean Value Coordinates Based Methods

Mean Value Coordinates (MVC) were presented by Floater et al. [FKR05] in 2005 as a generalisation of the Classical Barycentric Coordinates to express a point inside a polyhedron as a convex combination of its vertices $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$. This polyhedron, which we will call *cage* from now on, has triangular facets and the coordinates are well-defined and positive in the interior of it, if it is convex, or in its kernel, if it is concave. Assuming the faces of the polyhedron, edges of a 2D polygon, as opaque walls, the kernel is the volume inside the polyhedron from which all its vertices can be viewed directly. Figure 3.1 shows the example of the kernel of a 2D polygon.

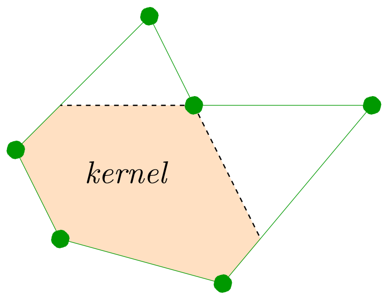


Figure 3.1: Kernel of a polygon.

In this section, we study a set of methodologies that, while using the same principles as the MVC to compute the coordinates, try to free this computation from a fixed and static cage. First of all, let us explain the MVC in depth.

The basic idea to derive the MVC is the observation that the integral of all unit normals $\vec{\mathbf{n}}$ around a sphere S is zero

$$\int_S \vec{\mathbf{n}} = \vec{\mathbf{0}}. \tag{3.4}$$

Given a point $\mathbf{p}$ inside the cage, its triangular boundary $\mathcal{T}$ is projected on the unit sphere centered on $\mathbf{p}$, $\mathcal{S}_{\mathbf{p}}$. We can see this projection as a partition of the sphere with spherical tetrahedra. Each spherical tetrahedron is composed of a projected triangle $\widehat{T} = [\widehat{\mathbf{v}}_i, \widehat{\mathbf{v}}_j, \widehat{\mathbf{v}}_k]$ of the cage with the point $\mathbf{p}$, as shown in Figure 3.2. Thus, the integral in Equation (3.4) can be written as

$$\int_{\mathcal{S}_{\mathbf{p}}} \vec{\mathbf{n}} = \sum_{T \in \mathcal{T}} \int_{\widehat{T}} \frac{(\mathbf{v} - \mathbf{p})}{d} = \vec{\mathbf{0}}, \tag{3.5}$$

where $\mathbf{v}$ is a point on the spherical triangle $\widehat{T}$ and $d = \|(\mathbf{v} - \mathbf{p})\|$.

Let us define the vectors $\vec{\mathbf{q}} = \frac{(\mathbf{v}-\mathbf{p})}{d}$, $\vec{\mathbf{q}}_i = \frac{(\mathbf{v}_i-\mathbf{p})}{d_i}$, $\vec{\mathbf{q}}_j = \frac{(\mathbf{v}_j-\mathbf{p})}{d_j}$ and $\vec{\mathbf{q}}_k = \frac{(\mathbf{v}_k-\mathbf{p})}{d_k}$, where $d_i = \|(\mathbf{v}_i - \mathbf{p})\|$, $d_j = \|(\mathbf{v}_j - \mathbf{p})\|$ and $d_k = \|(\mathbf{v}_k - \mathbf{p})\|$. Then, $\vec{\mathbf{q}}$ is expressed as the convex combination $\vec{\mathbf{q}} = \beta_i \vec{\mathbf{q}}_i + \beta_j \vec{\mathbf{q}}_j + \beta_k \vec{\mathbf{q}}_k$, where the values $(\beta_i, \beta_j, \beta_k)$ are the spherical barycentric coordinates of $\vec{\mathbf{q}}$ with respect to $\vec{\mathbf{q}}_i$, $\vec{\mathbf{q}}_j$ and $\vec{\mathbf{q}}_k$. Hence,

$$\beta_i = \frac{vol(\vec{\mathbf{q}}, \vec{\mathbf{q}}_j, \vec{\mathbf{q}}_k)}{vol(\vec{\mathbf{q}}_i, \vec{\mathbf{q}}_j, \vec{\mathbf{q}}_k)}, \quad \beta_j = \frac{vol(\vec{\mathbf{q}}_i, \vec{\mathbf{q}}, \vec{\mathbf{q}}_k)}{vol(\vec{\mathbf{q}}_i, \vec{\mathbf{q}}_j, \vec{\mathbf{q}}_k)} \quad \text{and} \quad \beta_k = \frac{vol(\vec{\mathbf{q}}_i, \vec{\mathbf{q}}_j, \vec{\mathbf{q}})}{vol(\vec{\mathbf{q}}_i, \vec{\mathbf{q}}_j, \vec{\mathbf{q}}_k)}$$

are ratios of the volumes of the tetrahedra, which are computed like $vol(\vec{\mathbf{q}}_i, \vec{\mathbf{q}}_j, \vec{\mathbf{q}}_k) = \left|\frac{|\vec{\mathbf{q}}_i \cdot (\vec{\mathbf{q}}_j \times \vec{\mathbf{q}}_k)|}{6}\right|$. Redefining Equation (3.5)

$$\begin{aligned}
\sum_{T \in \mathcal{T}} \int_{\widehat{T}} \frac{(\mathbf{v} - \mathbf{p})}{d} &= \sum_{T \in \mathcal{T}} \int_{\widehat{T}} \beta_i \vec{\mathbf{q}}_i + \beta_j \vec{\mathbf{q}}_j + \beta_k \vec{\mathbf{q}}_k \\
&= \sum_{T \in \mathcal{T}} \left(\vec{\mathbf{q}}_i \int_{\widehat{T}} \beta_i + \vec{\mathbf{q}}_j \int_{\widehat{T}} \beta_j + \vec{\mathbf{q}}_k \int_{\widehat{T}} \beta_k \right) \\
&= \sum_{T \in \mathcal{T}} \beta_{i,\widehat{T}} \vec{\mathbf{q}}_i + \beta_{j,\widehat{T}} \vec{\mathbf{q}}_j + \beta_{k,\widehat{T}} \vec{\mathbf{q}}_k.
\end{aligned} \tag{3.6}$$

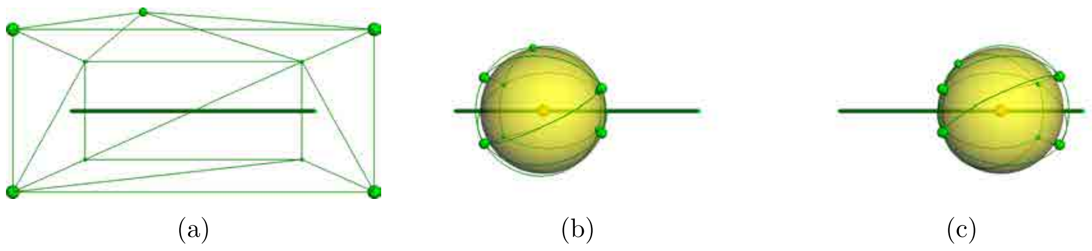


Figure 3.2: Partition of the unit sphere $\mathcal{S}_{\mathbf{p}}$ centered on $\mathbf{p}$ as the result of the projection of the boundary of the cage on it. (a) shows a model of a line enclosed in a cage of nine control points. (b) and (c) display two different projections of the deformation handles and the cage over the spheres $\mathcal{S}_{\mathbf{p}_1}$ and $\mathcal{S}_{\mathbf{p}_2}$. The points $\mathbf{p}_1$ and $\mathbf{p}_2$ are those in orange.

Figure 3.3 shows the nomenclature we use to derive the MVC. The vector $\vec{\mathbf{n}}_{\widehat{T}}$ is the sum of the normals in the spherical face of the tetrahedron, that is $\vec{\mathbf{n}}_{\widehat{T}} = \int_{\widehat{T}} \vec{\mathbf{q}}$. Vectors $\vec{\mathbf{n}}_{ij}$, $\vec{\mathbf{n}}_{jk}$ and $\vec{\mathbf{n}}_{ki}$ are the normals of the planar faces $F_{ij} = [\widehat{\mathbf{v}}_i, \mathbf{p}, \widehat{\mathbf{v}}_j]$, $F_{jk} = [\widehat{\mathbf{v}}_j, \mathbf{p}, \widehat{\mathbf{v}}_k]$ and $F_{ki} = [\widehat{\mathbf{v}}_k, \mathbf{p}, \widehat{\mathbf{v}}_i]$, respectively, so $\vec{\mathbf{n}}_{ij} = \frac{\vec{\mathbf{q}}_i \times \vec{\mathbf{q}}_j}{\|\vec{\mathbf{q}}_i \times \vec{\mathbf{q}}_j\|}$, $\vec{\mathbf{n}}_{jk} = \frac{\vec{\mathbf{q}}_j \times \vec{\mathbf{q}}_k}{\|\vec{\mathbf{q}}_j \times \vec{\mathbf{q}}_k\|}$ and $\vec{\mathbf{n}}_{ki} = \frac{\vec{\mathbf{q}}_k \times \vec{\mathbf{q}}_i}{\|\vec{\mathbf{q}}_k \times \vec{\mathbf{q}}_i\|}$.

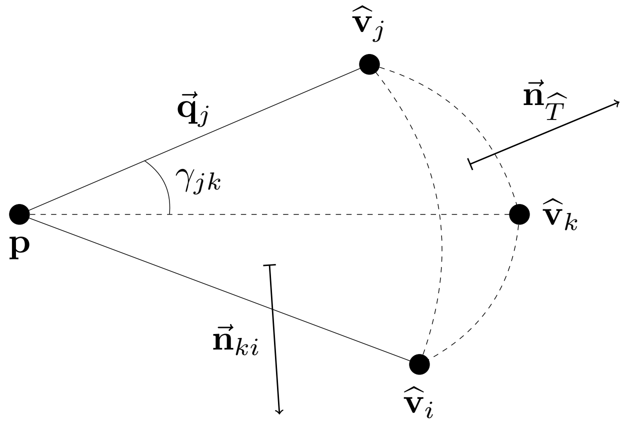


Figure 3.3: Nomenclature in a particular spherical tetrahedron $\widehat{T}$.

As stated before, it is a well-known fact that the normal integration over the surface of any closed body, and over a sphere particularly, is $\vec{\mathbf{0}}$. That means $\vec{\mathbf{n}}_{\widehat{T}} = a_{ij}\vec{\mathbf{n}}_{ij} + a_{jk}\vec{\mathbf{n}}_{jk} + a_{ki}\vec{\mathbf{n}}_{ki}$, where a_{ij}, a_{jk} and a_{ki} are the areas of the faces F_{ij}, F_{jk} and F_{ki}, respectively. The projection sphere is unitary, meaning its radius is equal to 1, so, the area a_{ij} is computed as $a_{ij} = \frac{\gamma_{ij}}{2}$, where γ_{ij} is the angle between $\vec{\mathbf{q}}_i$ and $\vec{\mathbf{q}}_j$ (the same for the areas a_{jk} and a_{ki}). Finally, the normal $\vec{\mathbf{n}}_{\widehat{T}}$ can be expressed as

$$\vec{\mathbf{n}}_{\widehat{T}} = \frac{\gamma_{ij}}{2}\vec{\mathbf{n}}_{ij} + \frac{\gamma_{jk}}{2}\vec{\mathbf{n}}_{jk} + \frac{\gamma_{ki}}{2}\vec{\mathbf{n}}_{ki}.$$

Then, the spherical barycentric coordinates of $\vec{\mathbf{n}}_{\widehat{T}}$ are

$$\begin{array}{lclcl} \beta_{i,\widehat{T}} & = & \frac{vol(\vec{\mathbf{n}}_{\widehat{T}},\vec{\mathbf{q}}_j,\vec{\mathbf{q}}_k)}{vol(\vec{\mathbf{q}}_i,\vec{\mathbf{q}}_j,\vec{\mathbf{q}}_k)} & = & \left|\frac{\vec{\mathbf{n}}_{\widehat{T}}\cdot(\vec{\mathbf{q}}_j\times\vec{\mathbf{q}}_k)}{\vec{\mathbf{q}}_i\cdot(\vec{\mathbf{q}}_j\times\vec{\mathbf{q}}_k)}\right| & = & \frac{\gamma_{ij}\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk}+\gamma_{jk}+\gamma_{ki}\vec{\mathbf{n}}_{ki}\cdot\vec{\mathbf{n}}_{jk}}{2\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk}} \\ \beta_{j,\widehat{T}} & = & \frac{vol(\vec{\mathbf{n}}_{\widehat{T}},\vec{\mathbf{q}}_i,\vec{\mathbf{q}}_k)}{vol(\vec{\mathbf{q}}_j,\vec{\mathbf{q}}_i,\vec{\mathbf{q}}_k)} & = & \left|\frac{\vec{\mathbf{n}}_{\widehat{T}}\cdot(\vec{\mathbf{q}}_i\times\vec{\mathbf{q}}_k)}{\vec{\mathbf{q}}_j\cdot(\vec{\mathbf{q}}_i\times\vec{\mathbf{q}}_k)}\right| & = & \frac{\gamma_{ij}\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{ki}+\gamma_{jk}\vec{\mathbf{n}_{jk}}\cdot\vec{\mathbf{n}}_{ki}+\gamma_{ki}}{2\vec{\mathbf{q}}_j\cdot\vec{\mathbf{n}}_{ki}} \\ \beta_{k,\widehat{T}} & = & \frac{vol(\vec{\mathbf{n}}_{\widehat{T}},\vec{\mathbf{q}}_i,\vec{\mathbf{q}}_j)}{vol(\vec{\mathbf{q}}_k,\vec{\mathbf{q}}_i,\vec{\mathbf{q}}_j)} & = & \left|\frac{\vec{\mathbf{n}}_{\widehat{T}}\cdot(\vec{\mathbf{q}}_i\times\vec{\mathbf{q}}_j)}{\vec{\mathbf{q}}_k\cdot(\vec{\mathbf{q}}_i\times\vec{\mathbf{q}}_j)}\right| & = & \frac{\gamma_{ij}+\gamma_{jk}\vec{\mathbf{n}_{jk}}\cdot\vec{\mathbf{n}}_{ij}+\gamma_{ki}\vec{\mathbf{n}_{ki}}\cdot\vec{\mathbf{n}}_{ij}}{2\vec{\mathbf{q}}_k\cdot\vec{\mathbf{n}}_{ij}}. \end{array} \tag{3.7}$$

As the volume is always positive, it does not matter the order in which the vectors $\vec{\mathbf{n}_{\widehat{T}}}$, $\vec{\mathbf{q}}_i$, $\vec{\mathbf{q}}_j$ and $\vec{\mathbf{q}}_k$ are chosen.

Next, Equation (3.6) is grouped by control points instead of triangles

$$\sum_{T\in\mathcal{T}} \beta_{i,\widehat{T}}\vec{\mathbf{q}}_i + \beta_{j,\widehat{T}}\vec{\mathbf{q}}_j + \beta_{k,\widehat{T}}\vec{\mathbf{q}}_k = \sum_{i=1}^{n}\sum_{T\ni\mathbf{v_i}} \beta_{i,\widehat{T}}\vec{\mathbf{q}}_i = \vec{\mathbf{0}},$$

and the expression of $\vec{\mathbf{q}}_i$ is also substituted

$$\sum_{i=1}^{n}\sum_{T\ni\mathbf{v_i}} \beta_{i,\widehat{T}}\vec{\mathbf{q}}_i = \sum_{i=1}^{n}\sum_{T\ni\mathbf{v_i}} \beta_{i,\widehat{T}}\frac{(\mathbf{v}_i-\mathbf{p})}{d_i} = \sum_{i=1}^{n}\omega_i(\mathbf{v}_i-\mathbf{p}).$$

The final Mean Value Coordinates $A = \{\alpha_1, \dots, \alpha_n\}$ are

$$\alpha_i = \frac{\omega_i}{\sum_{j=1}^{n}\omega_j}, \tag{3.8}$$

which satisfy the convex combination $\mathbf{p} = \sum_{i=1}^{n}\alpha_i\mathbf{v}_i$.

The MVC as explained can be used to perform *space deformations.* They are noted by their simplicity and efficiency as they are computed using a closed and simple formula. However, the MVC are a system of coordinates thought to be used in *cage-based* deformations and, as any cage-based system, they inherit their main disadvantages:

1. The user needs to specify the connectivity of the cage manually.

2. Two cages with exactly the same control points but distinct connectivity result in completely different sets of coordinates, although the difference between connectivities may be just a single edge.

Our goal is to derive a system of coordinates that takes advantage of the MVC benefits while trying to solve the problems defined by the cage-based systems. For this reason, our first attempts are completely based on the MVC coordinates and only try to define an automatic connectivity between the user-defined control points to support the computation of these coordinates. In addition, we want this connectivity to be computed from the point of view of $\mathbf{p}$, the point of the model to which the coordinates are computed. This ensures that $\mathbf{p}$ always lies inside the convex hull, or the kernel, of the automatically computed polyhedron. With these premises, we achieve an important benefit:

the user is freed from defining a cage and he or she only needs to locate a set of control points around the model[1].

The following sections are devoted to present the development of these approaches and to discuss their advantages and disadvantages.

3.1.1 Delaunay Triangulation Approach

The first idea we studied was to perform a Delaunay Triangulation. Given the point $\mathbf{p}$ of the model and the control points $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, the Delaunay Triangulation Approach (DTA) firstly projects the control points over the unit sphere $\mathcal{S}_\mathbf{p}$ centered on $\mathbf{p}$. Then, a Delaunay Triangulation is performed, so we get a triangular mesh projected on the surface of $\mathcal{S}_\mathbf{p}$. The reader can imagine it as the star constellations on the celestial dome viewed from $\mathbf{p}$.

This Delaunay Triangulation is the supporting structure to compute the DTA coordinates, which are completely based on the premises and development of the MVC as explained in the introduction of this section.

The DTA coordinates inherit the simplicity of the MVC. Moreover, the DTA does not need the control points to be located in a convex hull, so the user has complete freedom to locate the handles wherever he or she considers more suitable. However, the DTA triangulates these control points depending on their projection over the sphere of a particular point $\mathbf{p}$. This means that the triangulation is not constant for all the points of the model: two neighbouring points $\mathbf{p}_1$ and $\mathbf{p}_2$ can produce two different Delaunay Triangulations because of the flips on certain edges, as shown in Figure 3.4.

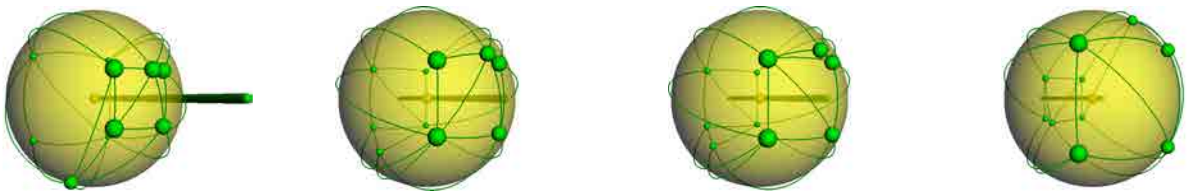


Figure 3.4: Example of flips appearing on adjacent Delaunay Triangulations. This example shows the projection of the control points over a sphere and their Delaunay Triangulation for different points of the model. The model is composed of a line of points and the reader can observe the evolution of the Delaunay Triangulation and how edges flip when we travel through the points of the model.

Previously, we have explained that the MVC computation totally depends on the triangulation defined over the control points. As the DTA coordinates are completely based on the MVC, they also inherit this dependency. This means that the functions of the coordinates show discontinuities whenever a flip occurs between adjacent triangulations[2]. Note that this situation can be compared with the technique presented in [LKCOL07]. Positive Mean Value Coordinates also present discontinuities in the vicinity of the concave parts of the cages.

[1]Notice that the region the user wants to deform must be enclosed inside the convex hull of the control points.

[2]Adjacent triangulations are those that correspond to two different model points $\mathbf{p}_1$ and $\mathbf{p}_2$ that are very close in the euclidean space.

The discontinuities on our functions also arise during the deformation process, resulting in artifacts and rough or unpleasant deformations of the models. Figure 3.5 shows that the DTA coordinates are not suitable to use in spatial deformation techniques.

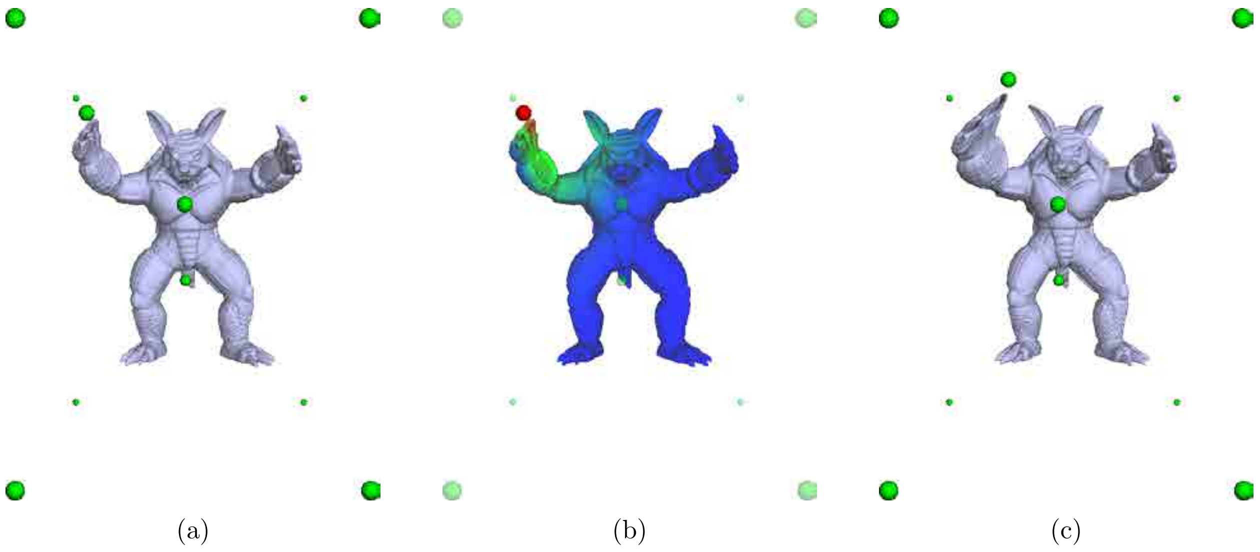


Figure 3.5: Example of a deformation of the *Armadillo* model by means of the DTA coordinates. In the initial situation, the *Armadillo* is enclosed in a cubic convex hull with three control points inside it, as shown in (a). Then, (b) shows the heat map that corresponds to the red deformation handle. The area in red is the most influenced by the control point. In contrast, the area in blue remains almost constant when this control point is dragged. Finally, (c) shows a deformation caused by the translation of the selected handle.

When facing this problem, we needed to think of some solution to correct the discontinuities caused by the flips on the triangulations. The idea we considered was to apply a blending function over the coordinate functions. This blending function would be applied whenever we dealt with different adjacent triangulations in order to smooth the resulting coordinates.

Bézier Blending

Given two adjacent triangles T_1 and T_2, we can detect when a flip is about to take place analysing the angles γ and δ corresponding to the nonadjacent vertices between them (see Figure 3.6). While $\gamma+\delta \leq 180°$, the triangulation is balanced and the created triangles $T_1' = [\mathbf{a}, \mathbf{b}, \mathbf{c}]$ and $T_2' = [\mathbf{a}, \mathbf{b}, \mathbf{d}]$ keep constant. However, once the sum $\gamma + \delta = 180°$ is reached, the flip takes place and the two triangles become $T_1'' = [\mathbf{a}, \mathbf{c}, \mathbf{d}]$ and $T_2'' = [\mathbf{b}, \mathbf{c}, \mathbf{d}]$.

We want to relax the discontinuities caused by flips by applying a blending function whenever a particular triangulation approaches a flip, that is, when $\gamma + \delta = 180° - \epsilon$. The constant ϵ allows us to control the area of the model whose points will have coordinates affected by the blending function. The bigger the value of ϵ is, the bigger the impact of the blending function over the coordinates will be.

Then, given a point $\mathbf{p}$, we test if any pair of adjacent triangles T_1 and T_2 of the induced Delaunay Triangulation is near a flip. If so, we compute two sets of DTA coordinates $A' = \{\alpha_1', \ldots, \alpha_2'\}$ and $A'' = \{\alpha_1'', \ldots, \alpha_2''\}$. We use the triangles T_1' and T_2' to compute the coordinates A' and the triangles T_1'' and T_2'' to compute A''. Finally, the final set of coordinates $A = \{\alpha_1, \ldots, \alpha_2\}$ results from an

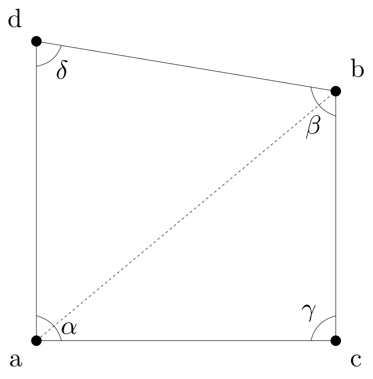


Figure 3.6: Two adjacent triangles and their angles.

interpolation between A' and A''

$$A = (1-\mu)A' + \mu A''.$$

We studied the use of a cubic Bézier curve $\mathcal{B}$ to compute the blending weight μ. The control points for $\mathcal{B}$ are $P_0 = (-\epsilon, 1)$, $P_1 = (1-\epsilon, 1)$, $P_2 = (\epsilon - 1,)$ and $P_3 = (\epsilon, 0)$, so μ is given by

$$\mu = -\epsilon(1-x)^3 + 3(1-\epsilon)(1-x)^2 x + 3(\epsilon-1)(1-x)x^2 + \epsilon x^3.$$

Figure 3.7 shows the cubic Bézier used.

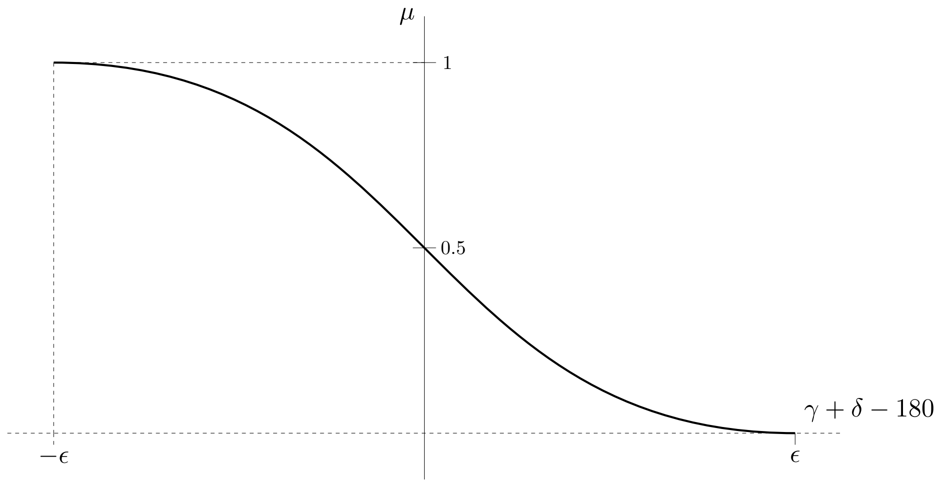


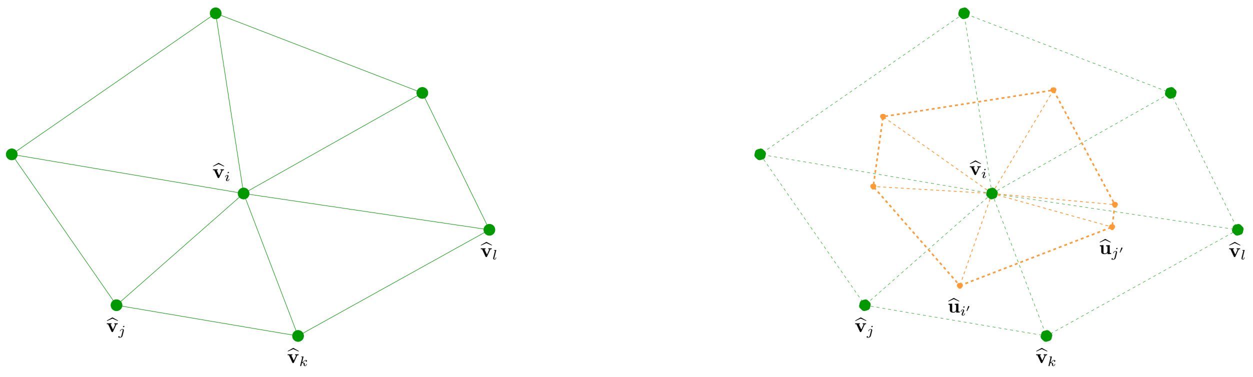
Figure 3.7: Bézier curve used to blend two consecutive Delaunay Triangulations with flips between their triangles.

Although the blending function allows us to reduce and smooth the effect of flips between adjacent Delaunay Triangulations, the deformations performed by means of these DTA coordinates are still rough and present sharp features. The coordinate functions are completely continuous. However, the blending area emerges during the deformation process and results in unpleasant distortions. Thus, neither are the blended DTA coordinates suitable enough for use in spatial deformation techniques.

3.1.2 Voronoi Regions Triangulation Approach

Seeing that a simple blending was not sufficient to hide the discontinuities of the DTA and to get a set of DTA coordinates to perform pleasant and intuitive deformations, we had to address the problem caused by the flips at its root. In other words, we needed to find another way to connect the control points that was free from flips and discontinuities between adjacent connections.

Each Delaunay Triangulation has associated the dual graph of the Voronoi Diagram. Unlike Delaunay Triangulations, which evolve discontinuously due to the flips, the associated Voronoi Diagrams are continuous. For this reason, we considered improving the DTA coordinates using the Voronoi Diagram to create the supporting triangulation for the coordinates computation, as shown in Figure 3.8. We called this experiment the Voronoi Regions Triangulation Approach (VRTA).



(a) Delaunay Triangulation (b) Voronoi Region Triangulation

Figure 3.8: Comparison between a Delaunay Triangulation and a Voronoi Region Triangulation.

The VRTA coordinates are based on the same principles as the DTA coordinates:

1. the main goal is to automatically compute a connectivity between the user-defined control points,
2. we want this connectivity to be defined from the point of view of the studied point $\mathbf{p}$ of the model and
3. the final computation of the VRTA coordinates is completely based on the MVC.

The only difference between the two methodologies lies in the triangulation used as the support structure for the computation of the coordinates. Given a point $\mathbf{p}$ of the model, the VRTA projects the control points over the unit sphere $\mathcal{S}_{\mathbf{p}}$ and compute their Voronoi Diagram. An easy way to do this is firstly to compute the Delaunay Triangulation of the projected control points. Then, the Voronoi vertices are computed as the circumcenters of the Delaunay triangles and projected over the sphere. The Voronoi vertex $\widehat{\mathbf{u}}_{i'}$ corresponds to the Delaunay triangle whose vertices are the control points $\widehat{\mathbf{v}}_i$, $\widehat{\mathbf{v}}_j$ and $\widehat{\mathbf{v}}_k$. Finally, the Voronoi vertices are connected to define the boundaries of the Voronoi regions and these are triangulated as shown in Figure 3.8(b).

Once the underlying structure is obtained, the process computes an intermediate set of coordinates $\Omega_V = \{\omega_{V,1}, \ldots, \omega_{V,n}, \omega_{V,1'}, \ldots, \omega_{V,l'}\}$ following the MVC development. Observe that the cardinality of Ω_V is $|\Omega_V| = n + l$, where n is the number of control points and l is the number of Voronoi vertices, all of them projected over the sphere $\mathcal{S}_{\mathbf{p}}$, so

$$\overbrace{\sum_{i=1}^{n} \omega_{V,i}\widehat{\mathbf{v}}_i}^{\text{Control Points}} + \overbrace{\sum_{i'=1}^{l} \omega_{V,i'}\widehat{\mathbf{u}}_{i'}}^{\text{Voronoi Vertices}} = \mathbf{0}.$$

The Voronoi vertices do not exist from the point of view of the user. That means that the weights applied to them must be transferred to the control points somehow, so that the points of the model are completely and solely defined by these control points. For this reason, we studied two different ways to do this transference, both of them based on defining the Voronoi vertices with respect to the control points by means of a system of coordinates:

- Spherical Barycentric Coordinates Transference: computes the Spherical Barycentric Coordinates (SBC) of the Voronoi vertices with respect to the control points that compose the corresponding Delaunay triangles.
- Natural Neighbour Coordinates Transference: computes the Natural Neighbour Coordinates (NNC) of the Voronoi vertices with respect to its neighbouring control points.

We explain each transference method below.

Spherical Barycentric Coordinates Transference

Given the Voronoi vertex $\widehat{\mathbf{u}}_{i'}$, that corresponds to the Delaunay triangle $\widehat{T} = [\widehat{\mathbf{v}}_i, \widehat{\mathbf{v}}_j, \widehat{\mathbf{v}}_k]$, we define $\widehat{\mathbf{u}}_{i'}$ with respect to $\widehat{\mathbf{v}}_i$, $\widehat{\mathbf{v}}_j$ and $\widehat{\mathbf{v}}_k$ by means of the Spherical Barycentric Coordinates (SBC) $(\beta_i, \beta_j, \beta_k)$: $\widehat{\mathbf{u}}_{i'} = \beta_i\widehat{\mathbf{v}}_i + \beta_j\widehat{\mathbf{v}}_j + \beta_k\widehat{\mathbf{v}}_k$. The same weights $(\beta_i, \beta_j, \beta_k)$ used to reproduce $\widehat{\mathbf{u}}_{i'}$ can also be used to interpolate values from $\widehat{\mathbf{u}}_{i'}$ to $\widehat{\mathbf{v}}_i$, $\widehat{\mathbf{v}}_j$ and $\widehat{\mathbf{v}}_k$. Therefore, the transference of the intermediate coordinate $\omega_{V,i'}$, corresponding to the Voronoi vertex $\widehat{\mathbf{u}}_{i'}$, is done interpolating its value in the control points, through the SBC $(\beta_i, \beta_j, \beta_k)$. Then, the coordinate ω_i for the control point $\widehat{\mathbf{v}}_i$ is

$$\omega_i = \omega_{V,i} + \sum_{\widehat{\mathbf{v}}_i \ni \widehat{T}} \beta_i \omega_{V,i'}.$$

The transference of the weights from the Voronoi vertices to the control points results in the set of coordinates $\Omega = \{\omega_1, \ldots, \omega_n\}$. This set Ω is the base to finally compute the coordinates $A = \{\alpha_1, \ldots, \alpha_n\}$ as in Equation (3.8).

Although the SBC of the Voronoi vertices are well-defined, a problem came up in our approach. The SBC computation relies on the Delaunay Triangulation of the control points. This fact makes that the discontinuities caused by flips arise again in the VRTA coordinates functions. These results made us conclude that the SBC are not suitable to transfer the weights from the Voronoi vertices to the control points. We needed to define a new method to do the transference that did not depend on the underlying Delaunay Triangulation.

Natural Neighbour Coordinates Transference

The Natural Neighbour Coordinates (NNC) [BC01] allow the definition of point p with respect to its neighbouring points in the 2D plane. This is done from the differences in the areas of the

Voronoi regions when the Voronoi Diagrams are computed considering, or not, p. So NNC do not take into account any connectivity between the study points. For this reason, we contemplated performing the transference of the weights from the Voronoi vertices to the control points by means of their NNC.

Given a Voronoi vertex $\widehat{\mathbf{u}}_{i'}$, firstly, we need to project the control points and the Voronoi vertex from the unit sphere $\mathcal{S}_{\mathbf{p}}$ to a plane $\mathcal{P}$, because, as mentioned above, the NNC are defined in the 2D plane. We chose $\mathcal{P}$ as the plane that contains $\widehat{\mathbf{u}}_{i'}$ and which normal is $\vec{\mathbf{n}}_{\mathcal{P}} = \widehat{\mathbf{u}}_{i'} - \mathbf{p}$. Notice that $\mathcal{P}$ is tangent to the sphere $\mathcal{S}_{\mathbf{p}}$ through $\widehat{\mathbf{u}}_{i'}$. Then, the control points are projected onto $\mathcal{P}$ through a polar projection, where the pole $\mathcal{U}$ is the opposite point to $\widehat{\mathbf{u}}_{i'}$ on $\mathcal{S}_{\mathbf{p}}$. We call $\bar{\mathbf{v}}_i$ the projected control points on $\mathcal{P}$. See Figure 3.9 for an example of the polar projection.

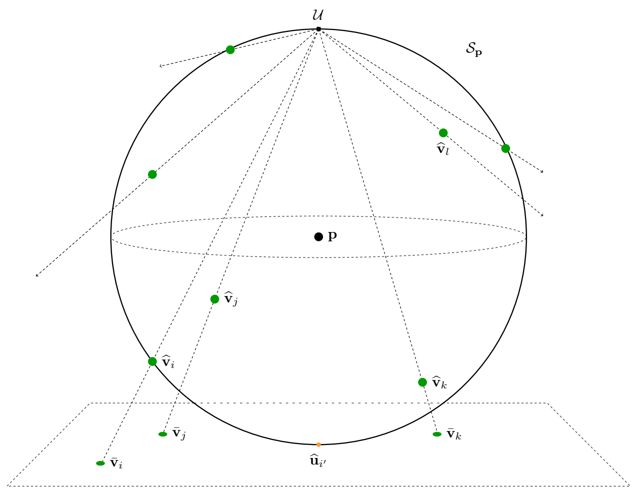


Figure 3.9: The control points are polar-projected into the plane $\mathcal{P}$, tangent to the unit sphere $\mathcal{S}_{\mathbf{p}}$ through $\widehat{\mathbf{u}}_{i'}$, to compute the NNC of the Voronoi vertex $\widehat{\mathbf{u}}_{i'}$.

Once the control points lie on $\mathcal{P}$, we compute the set $\bar{B} = \{\bar{\beta}_i, \ldots, \bar{\beta}_n\}$, which are the NNC of $\widehat{\mathbf{u}}_{i'}$ over $\mathcal{P}$, as in [BC01]. Finally, the set $\bar{B}$ of planar NNC is unprojected back to the sphere $\mathcal{S}_{\mathbf{p}}$ using the projection formulation development. So, we define

$$\bar{\mathbf{v}}_i = \widehat{\mathbf{v}}_i + \frac{\widehat{\mathbf{v}}_i - \mathcal{U}}{\|\widehat{\mathbf{v}}_i - \mathcal{U}\|} \cdot \|\bar{\mathbf{v}}_i - \widehat{\mathbf{v}}_i\|,$$

the projected control point on $\mathcal{P}$. Then, we substitute $\mathcal{U} = -\widehat{\mathbf{u}}_i$, so

$$\bar{\mathbf{v}}_i = \widehat{\mathbf{v}}_i + \frac{\widehat{\mathbf{v}}_i + \widehat{\mathbf{u}}_i}{\|\widehat{\mathbf{v}}_i + \widehat{\mathbf{u}}_i\|} \cdot \|\bar{\mathbf{v}}_i - \widehat{\mathbf{v}}_i\|$$

and call $\delta_i = \frac{\|\bar{\mathbf{v}}_i - \widehat{\mathbf{v}}_i\|}{\|\widehat{\mathbf{v}}_i + \widehat{\mathbf{u}}_i\|}$, so

$$\bar{\mathbf{v}}_i = \widehat{\mathbf{v}}_i(1 + \delta_i) + \widehat{\mathbf{u}}_{i'}\delta_i. \tag{3.9}$$

On the other hand, we know

$$\widehat{\mathbf{u}}_i = \sum_{i=1}^{n} \bar{\beta}_i \bar{\mathbf{v}}_i = \sum_{i=1}^{n} \beta_i \widehat{\mathbf{v}}_i, \tag{3.10}$$

where $\bar{\beta}_i$ are the NNC over the plane $\mathcal{P}$ and β_i are the NNC over the unit sphere $\mathcal{S}_{\mathbf{p}}$. By applying Equation (3.9) to Equation (3.10), we get the following mathematical development

$$\begin{aligned}
\widehat{\mathbf{u}}_{i'} &= \sum_{i=1}^{n} \bar{\beta}_i \left(\widehat{\mathbf{v}}_i(1+\delta_i) + \widehat{\mathbf{u}}_{i'}\delta_i\right) \\
&= \sum_{i=1}^{n} \bar{\beta}_i \widehat{\mathbf{v}}_i(1+\delta_i) + \widehat{\mathbf{u}}_{i'} \sum_{j=1}^{n} \bar{\beta}_j \delta_j \\
&= \frac{\sum_{i=1}^{n} \bar{\beta}_i \widehat{\mathbf{v}}_i(1+\delta_i)}{1 - \sum_{j=1}^{n} \bar{\beta}_j \delta_j} \\
&= \sum_{i=1}^{n} \widehat{\mathbf{v}}_i \frac{\bar{\beta}_i(1+\delta_i)}{1 - \sum_{j=1}^{n} \bar{\beta}_j \delta_j}.
\end{aligned}$$

In conclusion, the NNC of $\widehat{\mathbf{u}}_i$ over the unit sphere $\mathcal{S}_{\mathbf{p}}$ are the set of coordinates $B = \{\beta_i, \ldots, \beta_n\}$, where

$$\beta_i = \frac{\bar{\beta}_i(1+\delta_i)}{1 - \sum_{j=1}^{n} \bar{\beta}_j \delta_j}.$$

We use this set B of NNC to transfer the weights from the Voronoi vertices to the control points

$$\omega_i = \omega_{V,i} + \sum_{\widehat{\mathbf{v}}_i \ni \widehat{T}} \beta_i \omega_{V,i'},$$

so we get the set of coordinates $\Omega = \{\omega_1, \ldots, \omega_n\}$ and compute the final coordinates $A = \{\alpha_1, \ldots, \alpha_n\}$ as in Equation (3.8).

The VRTA coordinates with NNC transference solve the discontinuities induced by the flips on the Delaunay Triangulations. However, this transference is a cumbersome solution to our problem: the NNC can not be directly computed over the sphere and, furthermore, we need to compute the Voronoi Diagram for each Voronoi vertex $\widehat{\mathbf{u}}_{i'}$, which is an expensive process. In addition, we discovered a new kind of discontinuity that had gone unnoticed until then. These discontinuities, that we called *swords*, emerge when the point $\mathbf{p}$ of the model is collinear with two or more control points, as shown in Figure 3.10.

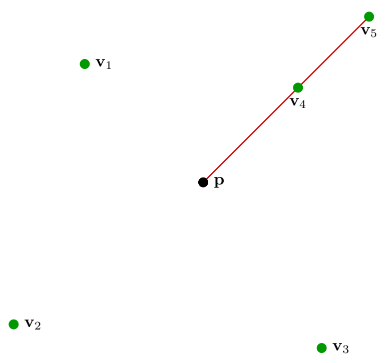


Figure 3.10: The point $\mathbf{p}$ is collinear with two control points. This situation causes a sword discontinuity in the VRTA coordinates.

When swords appear, the Voronoi Diagrams on the sphere do not evolve continuously between the model point $\mathbf{p}$ and its closest neighbours. In this situation, the collinear control points collapse in the same projection over the unit sphere of $\mathbf{p}$. When their projections are again separated for a neighbouring model point $\mathbf{p}'$, these control points swap part of their Voronoi regions in a rough way. Figure 3.11 presents a sequence that shows the situation before, during and after the sword appearance. This discontinuity has a direct impact over the VRTA and produces discontinuous coordinate functions that are not suitable for the deformation processes, as shown in Figure 3.12. Figure 3.13 shows a deformation process where the discontinuities on the VRTA coordinates arise.

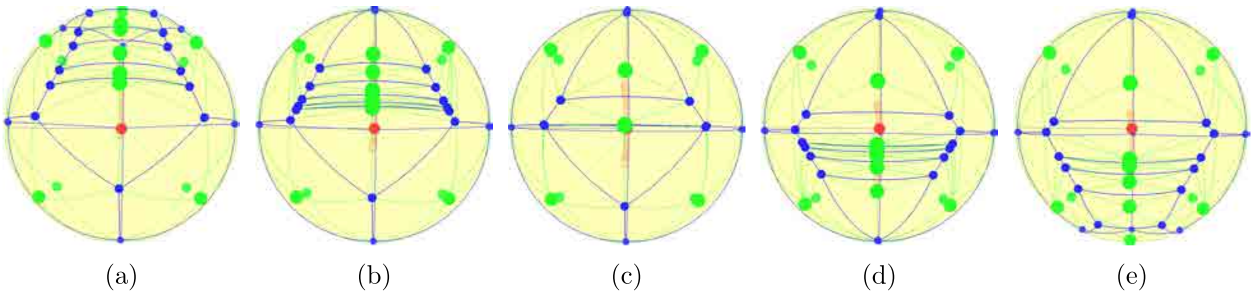


Figure 3.11: Example of a Voronoi Diagram sequence that shows the sword discontinuity. Note that the Voronoi regions and vertices, in blue, also collapse when the control points, in green, merge and the sword appears in (c). This sequence corresponds to the example presented in Figure 3.12.

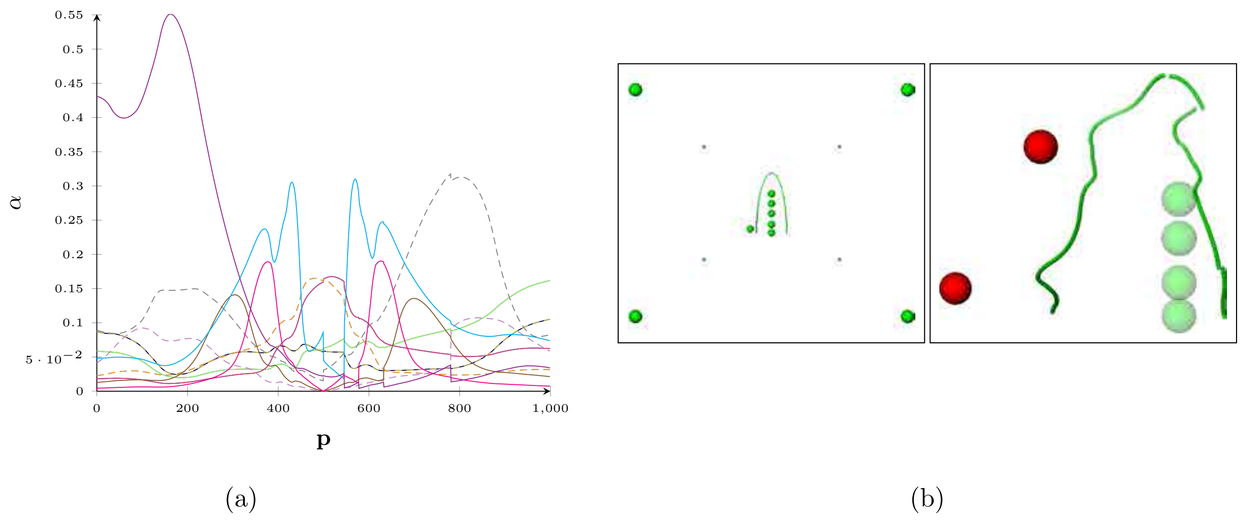


Figure 3.12: Figure (a) shows the VRTA coordinates with NNC transference functions corresponding to the initial configuration of a parabola surrounded by fourteen control points. The left and the right images in (b) show this initial configuration and a simple deformation produced by the translation of the control points in red, respectively. The parabola has been enclosed in a wide convex hull of control points to increase the visual effect of the discontinuities produced by the internal deformation handles. Moreover, the deformation image has been zoomed in for a better observation of the breakages.

We devised a solution that consisted in changing the classic Voronoi Diagram for the Laguerre Voronoi Diagram [Sug02], a weighted Voronoi Diagram over the sphere. The control points were weighted depending on their distance to the model point $\mathbf{p}$, so the closest control points had a bigger weight in the resulting Voronoi Diagram and the swords problem was solved. Finally, we also used the NNC transference to compute the final coordinates on the control points.

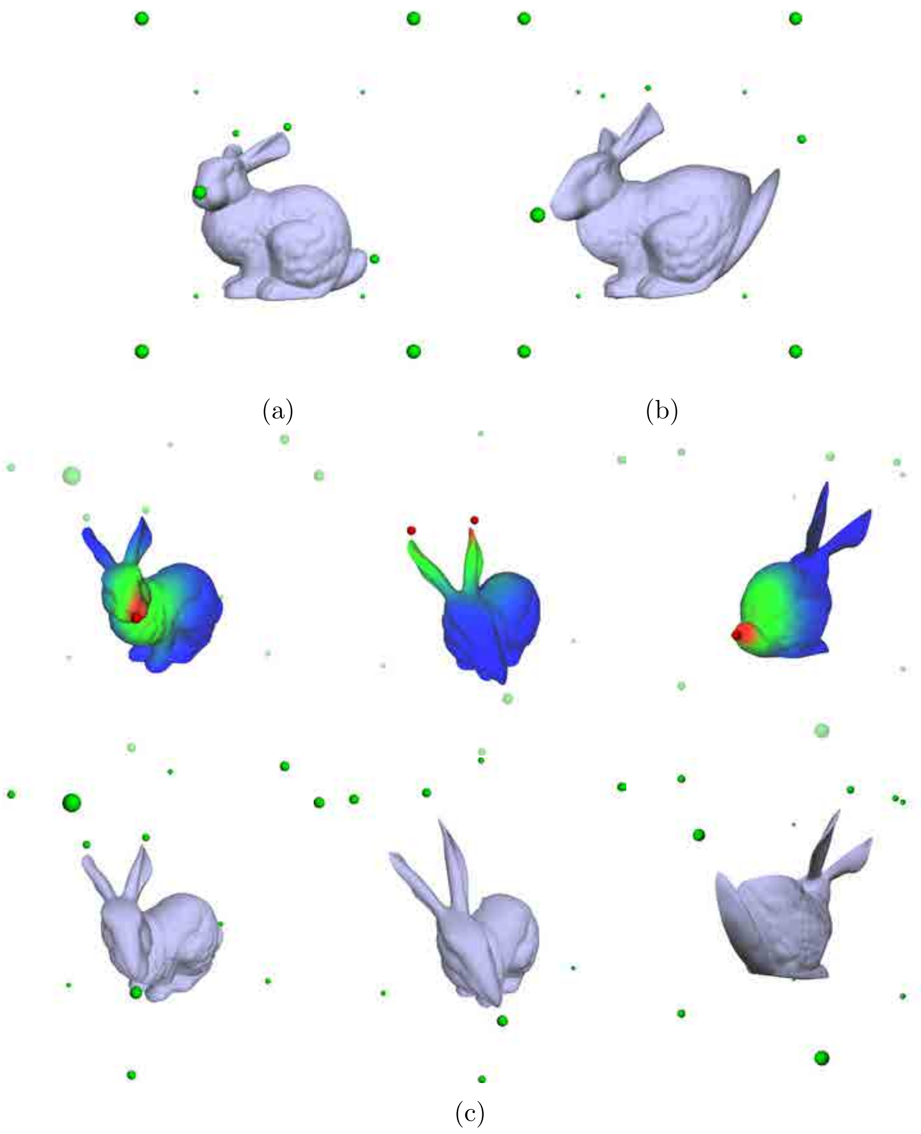


Figure 3.13: Example of a deformation process and the corresponding heat maps using the VRTA coordinates with NNC transference. On the one hand, (a) shows the initial configuration of the *Bunny* surrounded by twelve control points. On the other hand, (b) shows the final result of the deformation. Finally, (c) presents the sequence of the process. The images on the top illustrate the heat maps for the control points in red: areas in red are the most influenced by the handles and areas in blue remain almost constant when they are dragged. The images on the bottom show the modification caused on the *Bunny*.

Although combining the weighted Voronoi Diagram with the NNC transference technique gave correct, continuous and smooth results, we were not sufficiently satisfied due to the amount of Voronoi Diagrams this solution required. The technique was not efficient enough and the obtained results could not justify the usage of the new coordinates compared to the classic NNC of $\mathbf{p}$ with respect of the control points in the 3D space, which are more local and more efficient than our proposal.

3.1.3 Incremental Approach

The discontinuity problems in the VRTA coordinates can arise when there exists one or more control points that do not lie on their own convex hull, that is, there exist *internal* control points. However, VRTA behaves correctly when we only take into account the control points on the convex hull. In order to take advantage of this fact, we studied the Incremental Approach (IA). Given a point $\mathbf{p}$ of the model, the IA firstly computes its coordinates with respect to the control points on the convex hull. Then, it adds the rest of the handles one by one. Each time a new handle is added, this approach computes a new set of coordinates for $\mathbf{p}$ in an incremental way.

The coordinates of $\mathbf{p}$ with respect to the control points on the convex hull can be computed using any existing coordinates scheme, like the MVC, but also with the VRTA coordinates as explained in Section 3.1.2. So, in the first stage, we need to identify the control points in the set $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ that form the convex hull. In terms of simplicity, we suppose that the first l control points are these on the convex hull while the rest $m = n - l$ are internal to it. Then, we compute an initial set of coordinates $\Phi^l = \{\phi_1^l, \dots, \phi_l^l\}$ such that

$$\sum_{i=1}^{l} \phi_i^l \cdot \vec{\mathbf{q}}_i = \mathbf{0},$$

where $\vec{\mathbf{q}}_i$ are the convex hull control points projected over the unit sphere $\mathcal{S}_\mathbf{p}$ centered on $\mathbf{p}$. Then, we add the remaining internal m points one by one. In order to compute the set of coordinates Φ^t corresponding to the t-th control point addition, we firstly suppose $\phi_t^t = 1$, so

$$\sum_{i=1}^{t-1} \phi_i^t \cdot \vec{\mathbf{q}}_i + \phi_t^t \cdot \vec{\mathbf{q}}_t = \mathbf{0}$$

and consequently

$$\vec{\mathbf{q}}_t = -\sum_{i=1}^{t-1} \phi_i^t \cdot \vec{\mathbf{q}}_i. \tag{3.11}$$

On the other hand, we also want to define the coordinates Φ^t with respect to the previous set of coordinates Φ^{t-1} as a translation of these second ones

$$\phi_i^t = \phi_i^{t-1} + \vec{\mathbf{q}}_i \cdot \vec{\mathbf{d}} + k, \tag{3.12}$$

where $\vec{\mathbf{d}}$ is a unit vector and $k \geq 1$ is a constant value. Notice that, as far as $\vec{\mathbf{q}}_i$ and $\vec{\mathbf{d}}$ are unit vectors their dot product takes values in the range $[-1.0, 1.0]$. Then, $\vec{\mathbf{q}}_i \cdot \vec{\mathbf{d}} + k$ is in the range $[k - 1.0, k + 1.0]$, so the new coordinates ϕ_i^t are supposed to be positive, since ϕ_i^{t-1} are positive. Let us now develop Equations (3.11) and (3.12)

$$\begin{aligned} -\vec{\mathbf{q}}_t &= \sum_{i=1}^{t-1} \phi_i^t \cdot \vec{\mathbf{q}}_i \\ &= \sum_{i=1}^{t-1} (\phi_i^{t-1} + \vec{\mathbf{q}}_i \cdot \vec{\mathbf{d}} + k) \cdot \vec{\mathbf{q}}_i \\ &= Q \cdot \vec{\mathbf{d}} + \vec{\mathbf{f}}. \end{aligned} \tag{3.13}$$

The matrix Q and the vector $\vec{\mathbf{f}}$ are

$$
\begin{aligned}
Q &= \begin{pmatrix} \sum_{i=1}^{n-1} q_{ix}q_{ix} & \sum_{i=1}^{n-1} q_{ix}q_{iy} & \sum_{i=1}^{n-1} q_{ix}q_{iz} \\ \sum_{i=1}^{n-1} q_{ix}q_{iy} & \sum_{i=1}^{n-1} q_{iy}q_{iy} & \sum_{i=1}^{n-1} q_{iy}q_{iz} \\ \sum_{i=1}^{n-1} q_{ix}q_{iz} & \sum_{i=1}^{n-1} q_{iy}q_{iz} & \sum_{i=1}^{n-1} q_{iz}q_{iz} \end{pmatrix} \\
\vec{\mathbf{f}} &= k\sum_{i=1}^{n-1} \vec{\mathbf{q}}_i.
\end{aligned}
$$

Finally, we find the vector $\vec{\mathbf{d}}$ by solving the Equation System (3.13)

$$\vec{\mathbf{d}} = -Q^{-1} \cdot (\vec{\mathbf{q}}_t + \vec{\mathbf{f}}).$$

After all the control points have been added, the set of coordinates Φ^n is obtained and we can compute the final coordinates as

$$\mathbf{0} = \sum_{i=1}^{n} \phi_i^n \vec{\mathbf{q}}_i = \sum_{i=1}^{n} \phi_i^n \frac{\mathbf{v}_i - \mathbf{p}}{\|\mathbf{v}_i - \mathbf{p}\|},$$

so

$$\alpha_i = \frac{\frac{\phi_i^n}{\|\mathbf{v}_i - \mathbf{p}\|}}{\sum_{j=1}^{n} \frac{\phi_j^n}{\|\mathbf{v}_j - \mathbf{p}\|}}.$$

The resulting IA coordinates are well-defined when there are few interior control points, as shown in Figure 3.14. However, as we add more and more control points, the coordinates become negative as shown in Figure 3.15. This Figure shows an example using a model of a line of points surrounded with twenty-six control points. Notice that the plot of the functions of the coordinates presents negative values.

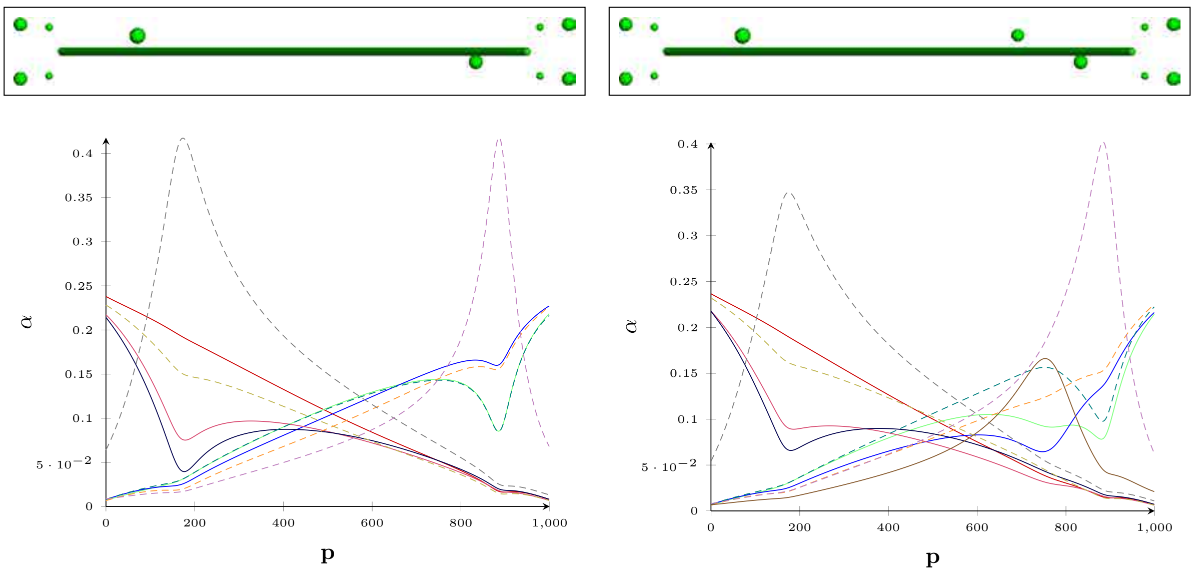


Figure 3.14: Example of the IA coordinates for the classical model of a line surrounded by ten control points, on the left, and eleven control points, on the right.

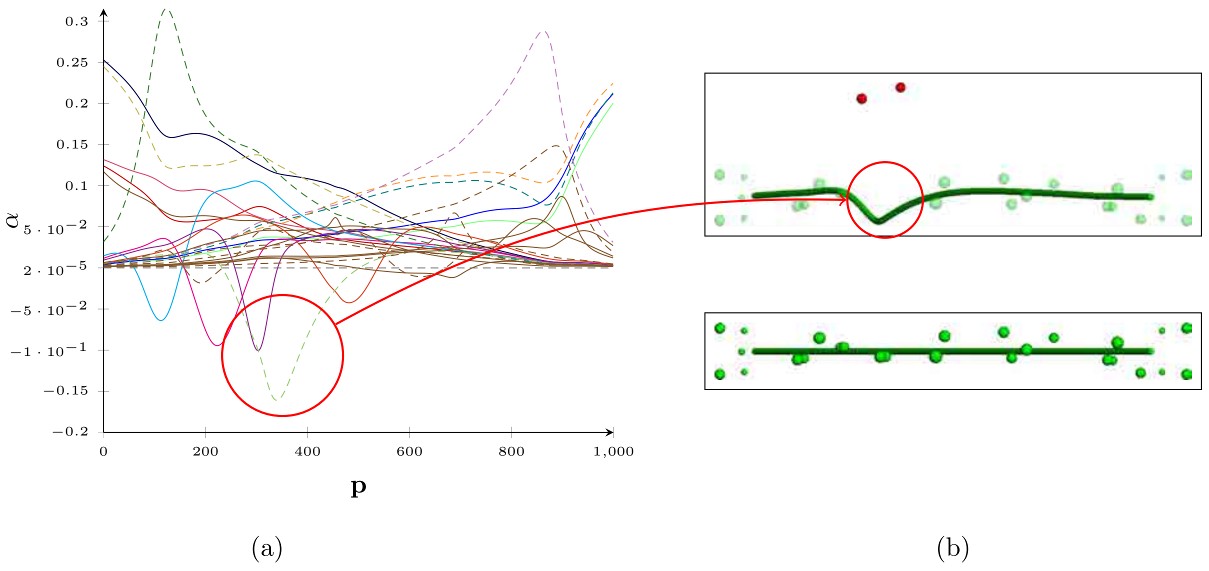


Figure 3.15: Example of the IA coordinates for the classical model of a line surrounded by twenty-six control points. Notice that the plot of the coordinates in (a) presents negative values. This situation causes unintuitive deformations: while the user drags some control points in a specific direction, some parts of the model are deformed in the opposite way, as observed in the upper part of (b). The lower image in (b) illustrates the initial configuration of the model and the deformation handles.

The negativity is due to our assumption that the vector $\vec{\mathbf{d}}$ is a unit vector, which led us to establish the constant value $k = 1$ in our experiments. However, we do not know the value of $\vec{\mathbf{d}}$ in advance, but it is computed through the Equation System (3.13). This means that we can not make any statement over the norm of $\vec{\mathbf{d}}$ because it is known after its computation. So the operation $\vec{\mathbf{q}}_i \cdot \vec{\mathbf{d}} + k$ in Equation (3.12), for $k = 1$, does not ensure the positivity of the incremental sets of coordinates Φ^t. A possible solution for this problem could be not to fix the value of k, but use the norm of the vector $\vec{\mathbf{d}}$ instead. That is, $k = \|\vec{\mathbf{d}}\|$.

Finally, this situation led us to further investigate how to compute a system of coordinates that fulfilled all of our requirements.

3.1.4 Conclusions

In this section we have studied some cage-free methods based on the MVC [FKR05]. We have described different techniques to automatically compute a supporting connectivity between the control points. However, as far as we define the connectivity of the control points from the point of view of the points of the model, the supporting structure is not constant through the model. This drawback, together with the high dependence of the MVC regarding the polyhedron formed by the control points, produces important discontinuities in the functions of the coordinates. These discontinuities render the techniques useless for deformation purposes.

On the other hand, the Laguerre Voronoi Diagram with NNC transference gives us good results. Nonetheless, its complexity makes us reject the method and we prefer a direct computation of the

NNC of the model with respect to the control points.

Finally, we have also studied an Incremental Approach to the problem. Nevertheless, further investigation on Equation (3.12) is required to guarantee the positiveness of the final coordinates.

Considering these results, we conclude that we need to study how to automatically compute the connectivity between the control points in a way that remains constant all over the model. This new direction leads us to the next section, where we analyse the idea of the layers to create a static and immutable structure between the control points.

3.2 Layered Based Methods

In Section 3.1, we have seen that there is not an efficient and elegant way to automatically compute a connectivity between the control points that depends on how these are seen from the point of view of $\mathbf{p}$, a point of the model. Some of the studied techniques do not make it possible to obtain a continuous set of coordinates inside the convex hull of the control points due to the changes of the supporting structure from point to point of the model. The other techniques, although resulting in a good set of coordinates, are not efficient enough and do not present any benefit compared to the existing schemes. Thus, we focused on changing our strategy in order to get a set of coordinates which satisfied all the properties of a well-defined system of coordinates, that is, a coordinate system that met all the properties exposed in the introduction of this chapter.

In this section, we study two new methodologies that build a layered structure to connect the control points. Given a subset $\mathcal{V}' \subseteq \mathcal{V}$ of the control points $\mathcal{V}$, we define a layer as the partition of the convex hull of $\mathcal{V}'$ into convex polyhedra. We can traverse all the polyhedra of a layer by crossing through their shared faces, as shown in Figure 3.16. Therefore, the supporting structure to compute the coodinates is composed of a set of polyhedra. This means that we require the coordinates to interpolate the faces of these polyhedra in order to reach the desired continuity property for our system.

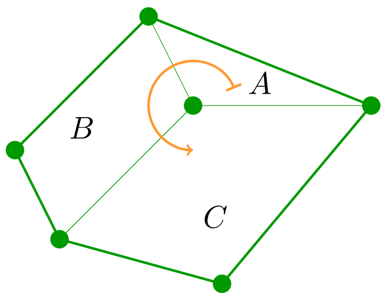


Figure 3.16: Example of a layer composed by three polyhedra: A, B and C. The arrow shows how these polyhedra can be traversed across their shared faces.

This layered structure is used for all the points of the model, so we work with the same supporting structure for all of them. With this fact, we try to avoid any discontinuity in the coordinate functions due to changes in the underlying connectivity from point to point of the model. In addition, the computation of a set of layers takes into account the control points located inside their own convex hull, in contrast to the triangulation done in the MVC based methods of Section 3.1. In this way, the layers structure the whole volume of this convex hull and not only its surface.

The following sections present two different methodologies to define a layered structure. They also explain how we compute the final coordinate system over the automatically created structure.

3.2.1 Numerical Integration Approach

An easy way to get a partition of the convex hull of the control points is to compute a tetrahedralisation. For this reason, the first idea we wanted to analyse was the behaviour of the MVC over the tetrahedralisation of the control points.

The process starts computing the Delaunay Tetrahedralisation of the handles, which is the supporting structure for the coordinates and is unique for all the points of the model. Then, given a particular point $\mathbf{p}$ of it, the method follows the same ideas and principles of the MVC [FKR05].

When the control points and their tetrahedralisation are projected over the unit sphere $\mathcal{S}_{\mathbf{p}}$ centered on $\mathbf{p}$, some tetrahedra may overlap with each other, as shown in Figure 3.17. This means that the MVC as proposed in [FKR05] can not be directly applied to our approach. Faced with this situation, we decided to perform a numerical integration to approximate the value of the MVC over our tetrahedralisation. We called this solution the Numerical Integration Approach (NIA).

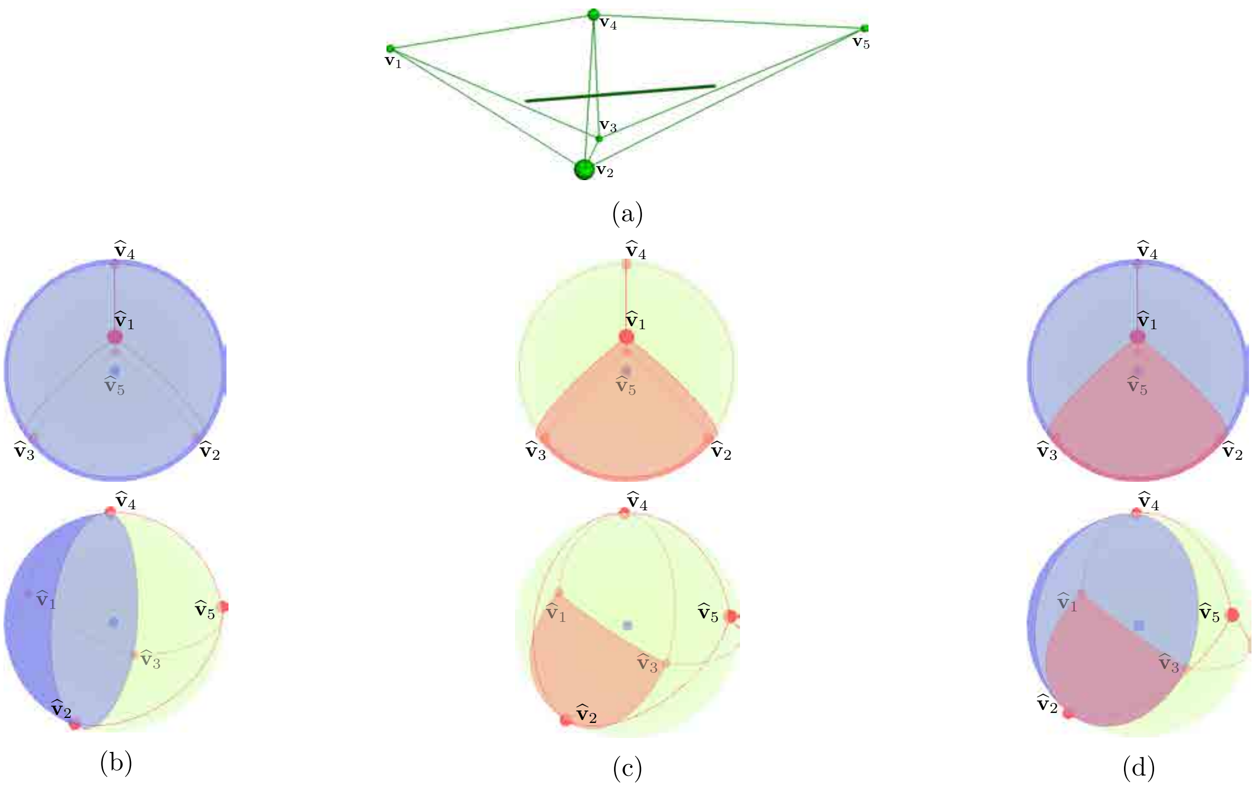
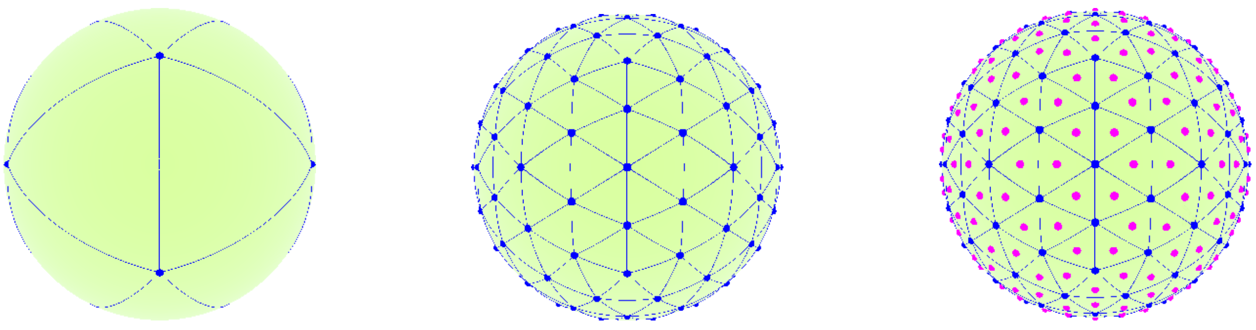


Figure 3.17: Tetrahedralisation of the control points projected over the unit sphere $\mathcal{S}_{\mathbf{p}}$. (a) shows the initial configuration. Then, for a particular model point $\mathbf{p}$, (b) and (c) highlight the spherical triangles $\widehat{T}_1 = (\widehat{\mathbf{v}}_2, \widehat{\mathbf{v}}_3, \widehat{\mathbf{v}}_4)$ and $\widehat{T}_2 = (\widehat{\mathbf{v}}_1, \widehat{\mathbf{v}}_2, \widehat{\mathbf{v}}_3)$. Finally, (d) emphasises the overlaped area in red.

The method pre-computes a set of sampling points p. These points lie over the unit sphere $\mathcal{S}_O$, which is centered on the origin of the euclidean 3D space. They are computed as the barycenters of the faces of a subdivided icosahedron projected over $\mathcal{S}_O$. The finer the subdivision is, the better the sphere will be sampled as more points would have been computed. Figure 3.18 shows the initial icosahedron, a 2-level subdivision of it and the sample points as the barycenters of the subdivided triangles.



(a) Spherical icosahedron.

(b) A 2-level subdivision of the spherical icosahedron.

(c) Sample points as the barycenters of the subdivided spherical triangles.

Figure 3.18: Example of a 2-level subdivided icosahedron and the sample points that represent the unit sphere surface.

The technique uses these sampling points to numerically integrate the coordinates over the unit spheres of the model points. They simply need to be translated from the origin sphere $\mathcal{S}_O$ to the particular point $\mathbf{p}$ sphere $\mathcal{S}_{\mathbf{p}}$.

On the other hand, each sampling point p represents a piece of the unit sphere surface. To be precise, each sample p corresponds to the area of the spherical triangle with which it is associated. This means that any value computed at p must be weighted by this area, which is

$$a_p = (\gamma_a + \gamma_b + \gamma_c - \pi)r^2,$$

where γ_a, γ_b and γ_c are the dihedral angles (see Figure 3.19) and r is the radius of the sphere, in our case, $r = 1$.

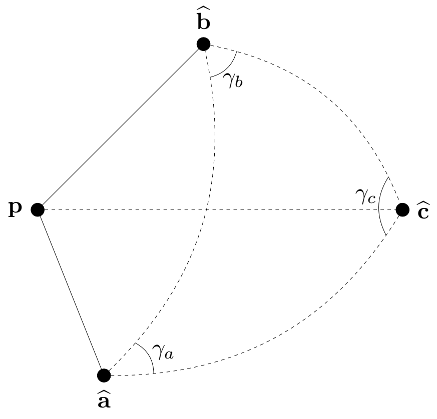


Figure 3.19: Dihedral angles of a spherical triangle.

Then, and going back to the particular case of the model point $\mathbf{p}$, the unit sphere $\mathcal{S}_{\mathbf{p}}$ is covered with the projected tetrahedralisation of the control points and sampled with the points p. Let us call $\widehat{\mathcal{T}} = \{\widehat{T}_1, \ldots, \widehat{T}_m\}$ the set of triangles of the tetrahedralisation projected over $\mathcal{S}_{\mathbf{p}}$. In order to compute the coordinates of $\mathbf{p}$ with respect to the control points, the NIA method uses the sampling points as the integration points. This means that, for every point p, the technique finds the triangles $\widehat{\mathcal{T}}_p \subset \widehat{\mathcal{T}}$ of the tetrahedralisation that contain p: $\widehat{\mathcal{T}}_p = \{\widehat{T}_l \in \widehat{\mathcal{T}} : p \in \widehat{T}_l\}$. Then, for each triangle $\widehat{T}_l = [\widehat{\mathbf{v}}_{i,l}, \widehat{\mathbf{v}}_{j,l}, \widehat{\mathbf{v}}_{k,l}] \in \widehat{\mathcal{T}}_p$, it computes the Spherical Barycentric Coordinates (SBC) of p with respect to the vertices of $\widehat{T}_l$, and weights them with the area a_p represented by p. The convex combination defining p is

$$p = a_p \sum_{\widehat{T}_l \in \widehat{\mathcal{T}}_p} (\beta_{i,l}\widehat{\mathbf{v}}_{i,l} + \beta_{j,l}\widehat{\mathbf{v}}_{j,l} + \beta_{k,l}\widehat{\mathbf{v}}_{k,l})\, w_l,$$

where $(\beta_{i,l}, \beta_{j,l}, \beta_{k,l})$ are the SBC of p and

$$w_l = \frac{vol(\widehat{\mathbf{v}}_{i,l}, \widehat{\mathbf{v}}_{j,l}, \widehat{\mathbf{v}}_{k,l}, \mathbf{p})^2}{\sum_{\widehat{T}_q \in \widehat{\mathcal{T}}_p} vol(\widehat{\mathbf{v}}_{i,q}, \widehat{\mathbf{v}}_{j,q}, \widehat{\mathbf{v}}_{k,q}, \mathbf{p})^2}$$

weights the influence of each triangle $\widehat{T}_l$ in the definition of p in a proportional way to the volume of the tetrahedron built by joining $\widehat{T}_l$ with the point $\mathbf{p}$. If we do not weight the SBC somehow, we are not able to reproduce p, but $|\widehat{\mathcal{T}}_p|p$.

Given a control point $\widehat{\mathbf{v}}_i$, the ω_i coordinate for it is

$$\omega_i = \sum_{\substack{\widehat{T}_l \in \widehat{\mathcal{T}} \\ \widehat{\mathbf{v}}_i \ni \widehat{T}_l}} w_l \beta_{i,l},$$

so

$$\sum_{i=1}^{n} \omega_i \widehat{\mathbf{v}}_i = \mathbf{0}.$$

Finally, the approach normalises $\Omega = \{\omega_i, \ldots, \omega_n\}$ to get the final coordinates $A = \{\alpha_i, \ldots, \alpha_n\}$

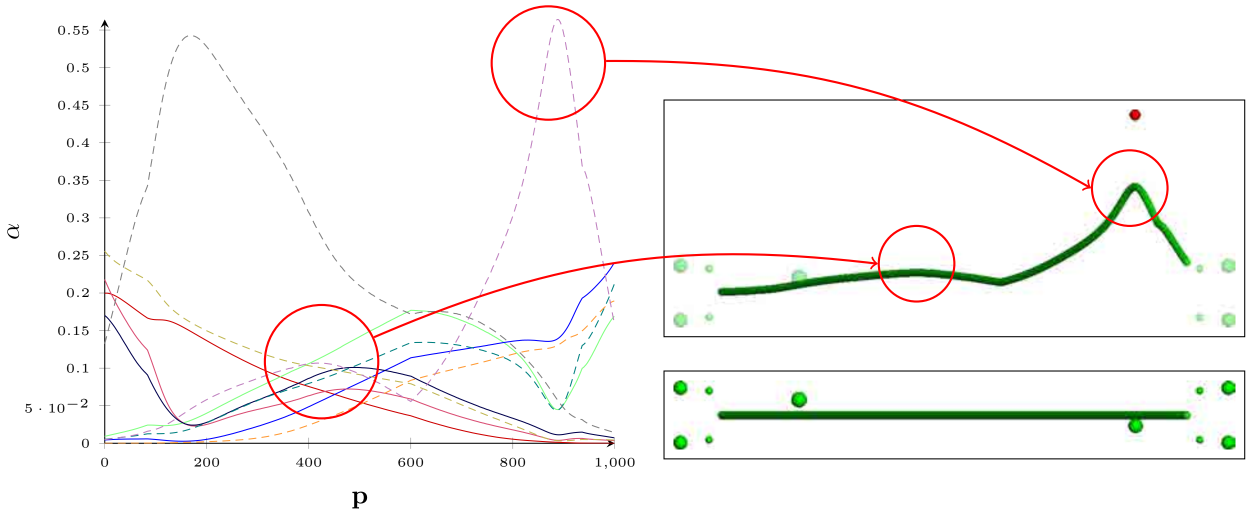
$$\alpha_i = \frac{\sum_{\substack{\widehat{T}_l \in \widehat{\mathcal{T}} \\ \widehat{\mathbf{v}}_i \ni \widehat{T}_l}} w_l \beta_{i,l}}{\sum_{j=1}^{n} \sum_{\substack{\widehat{T}_q \in \widehat{\mathcal{T}} \\ \widehat{\mathbf{v}}_j \ni \widehat{T}_q}} w_q \beta_{j,q}}.$$

This methodology gives us a set of continuous and positive NIA coordinates inside the convex hull of the control points. Figure 3.21 shows a deformation example using the NIA coordinates. It also presents the heat map of these coordinates over the model.

Nevertheless, although NIA coordinates are functional and can be used in real applications, they present some disadvantages:

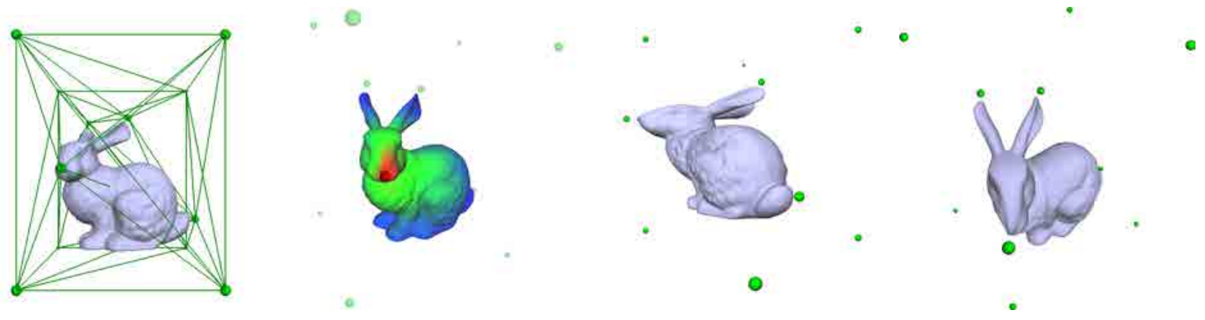
- They are computed through a discretisation of the sphere surface and a numerical integration. In other words, they do not have a closed formula. This means that they are not as efficient as other existing schemes.
- The functions of the coordinates present multiple local maxima. This situation implies that the deformation process is not as intuitive for the user as desired. Figure 3.20 shows the plot of the coordinates for a simple line model. Notice how the multiple local maxima affect the deformation of the line when a single control point is moved.

- The NIA coordinates inherit the C^0-continuity that the MVC present when the model point $\mathbf{p}$ lies on a triangle. This means that NIA coordinates are C^0-continuous across the faces of the tetrahedra. Therefore, the deformations around the regions where the model crosses a triangle produce sharp features and unpleasant results. This situation is shown in Figure 3.21.



(a) (b)

Figure 3.20: Figure (a) shows the NIA coordinate functions corresponding to the initial configuration of a line surrounded by ten control points. The bottom image in (b) presents the initial configuration while the top image shows a simple deformation produced by the translation of the handle in red. The red circles highlight some local maxima.



(a) (b)

Figure 3.21: In (a), we show the tetrahedralisation that supports the computation of the NIA coordinates. Notice that some of the triangular faces intersect the *Bunny* model. This fact causes C^0-continuity of the coordinate functions and a shear effect during the deformation process. This problem is illustrated in (b). First, it shows the heat map that corresponds to the control point near the nose of the *Bunny*: the red area is the most influenced by the handle and area in blue remains almost constant when it is dragged. Then, the next two images present two different views of the deformation. We can observe that the left side of the face of the *Bunny* is completely stretched.

For these reasons, we were not completely satisfied with the obtained results. With the Numerical Integration Approach, we had reached the desired continuity property for the coordinates. However, continuous coordinate functions are not enough to ensure intuitive and pleasant deformations. We needed further research in order to describe smoother systems of coordinates, more suitable for application in a deformation framework.

3.2.2 Catalog Approach

The NIA coordinates are not efficient or intuitive or smooth. However, their constant supporting structure over the control points gives them the important property of continuity. From this point, the Catalog Approach (CA) studies how to improve this structure in order to compute a set of well-defined coordinates that produces intuitive and smooth deformations efficiently.

The connectivity of the control points proposed by the CA consists in a set of layers, which we named *Catalog of Layers*, hence the name of the method. The first layer is always the convex hull of the deformation handles and it is intended to provide the required smoothness to the coordinate functions. The remaining layers are built by means of convex partitions of the initial convex hull, that is, each additional layer is a partition of the convex hull of the deformation handles. They can be defined in different ways, depending on two attributes:

- the kind of polyhedra used to achieve the required partitions;
- the number of *interior* control points taken into account in each layer.

Once the structure is defined, the coordinates for a given model point $\mathbf{p}$ result from the weighting of the coordinates of $\mathbf{p}$ with respect to the different layers.

The following sections are devoted to detailing the different methodologies to build the Catalog of Layers and the computation of the final CA coordinates. Furthermore, part of this work has been collected in [CVB13].

The Construction of the Catalog of Layers

In the previous section we have exposed that the CA builds a fixed set of layers to support the computation of the coordinates. We have settled that the first layer is the convex hull while the remaining ones are defined depending on the polyhedra used to construct the partitions and on the number of interior control points in each layer. Below, we study the different structures that we can achieve by the combination of these two attributes.

Control Points per Layer. Every time that we add a new layer to the catalog, the method firstly computes the convex hull of the control points. Then, it adds the remaining interior handles one by one. We distinguish between the layers that contain only a single interior control point and the layers that contain all the interior control points.

Independent Partitions. The Catalog of Layers is composed of $m + 1$ layers, where m is the number of interior control points. The first layer corresponds to the convex hull of the handles and each of the remaining ones is a convex partition of this convex driven by the corresponding interior handle. The first column in Figure 3.22 shows two examples of a catalog of independent partitions.

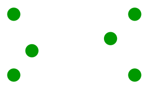
Nested Partitions. The Catalog of Layers is composed of two layers: the initial one, that represents the convex hull of the control points, and a second layer which is a partition of the convex hull taking into account all the interior control points. Hence, in order to build the second layer, the interior handles are added one by one, in the same order in which the user had defined them[3]. Every time a control point is added, the algorithm looks for the already existing convex that contains the new handle inside it. Once this convex is found, it is substituted by a convex partition that includes the new control point. The second column in Figure 3.22 shows two nested partitions.

Polyhedra Used in the Partitions. Once the next convex to be partitioned is located, it is split into a set of polyhedra. We distinguish the partitions depending on the kind of polyhedra used to split the old convexes: tetrahedra or arbitrary convex polyhedra.

Partitions of Tetrahedra. This is the simplest case and it consists in splitting the old convex into tetrahedra. Each tetrahedron is computed by joining the last inserted handle with each of the triangular faces of the initial convex, as shown in the first row in Figure 3.22.

Partitions of Arbitrary Convex Polyhedra. The method firstly splits the old convex into tetrahedra. Then, it traverses these tetrahedra and joins those adjacent polyhedra so that the new polyhedron remains convex, as shown in the second row in Figure 3.22.

Initial situation



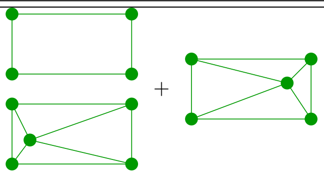
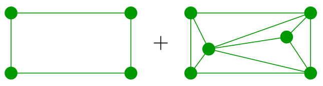
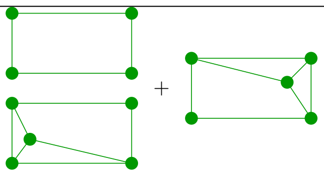
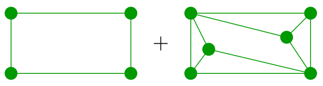
	Independent partitions	**Nested partitions**
Tetrahedra		
Arbitrary polyhedra		

Figure 3.22: Example of all the possible combinations to build the Catalog of Layers. The top part of the image shows the initial configuration of the deformation handles. The table below presents the layers that comprise the Catalog, depending on the relation between the partitions and the polyhedra used in them.

Computation of the Coordinates

Once the structure between the control points has been established, the CA computes the coordinates to relate a given model point **p** with the deformation handles. In this case, the system has to deal with multiple convexes which are arranged in several partitions of the convex hull of the control points, that is, in several layers. Each of these convexes contains a portion of the model to be deformed and each partition, or layer, of the structure contains the whole model inside it. This means that the coordinate system used must ensure the continuity between all the convexes in the

[3]The method assumes that the user defines the deformation handles from more general to more local impact on the later deformation.

same partition. This requirement implies that the computation of the coordinates on the faces of the convexes has to depend only on the control points defining these faces. In other words, the coordinates on a face must coincide for the left and right convexes of this face.

The method computes the coordinates of a model point $\mathbf{p}$ in three steps. First, it computes the set $\mathcal{C}_{\mathbf{p}}$, the *Convexes of* $\mathbf{p}$

$$\mathcal{C}_{\mathbf{p}} = \{C_j : \mathbf{p} \in C_j \wedge C_j \in \mathcal{L}_j, j \in \{1, l\}\},$$

where l is the number of layers of the catalog. That is, for each layer $\mathcal{L}_j$ of the structure, the technique finds the convex C_j that contains $\mathbf{p}$. If $\mathbf{p}$ lies on a face, it chooses one of the convexes that share this boundary. The second step consists in computing the set of coordinates of $\mathbf{p}$ with respect to each convex $C_j \in \mathcal{C}_{\mathbf{p}}$. Therefore,

$$\boldsymbol{\alpha}_j = (\alpha_{j,1}(\mathbf{p}), ..., \alpha_{j,n}(\mathbf{p})),$$

where n is the number of control points, are the coordinates of $\mathbf{p}$ with respect to the layer $\mathcal{L}_j$. If a handle $\mathbf{v}_i$ does not belong to C_j its corresponding coordinate $\alpha_{j,i}(\mathbf{p})$ is set to 0. Finally, the final set of coordinates is computed as a convex combination of the individual sets for each $C_j \in \mathcal{C}_{\mathbf{p}}$. That is,

$$\begin{array}{rcl}
\alpha_1(\mathbf{p}) & = & \sum_{j=1}^{l} \omega_j \alpha_{j,1}(\mathbf{p}) \\
\alpha_2(\mathbf{p}) & = & \sum_{j=1}^{l} \omega_j \alpha_{j,2}(\mathbf{p}) \\
 & \vdots & \\
\alpha_n(\mathbf{p}) & = & \sum_{j=1}^{l} \omega_j \alpha_{j,n}(\mathbf{p}),
\end{array}$$

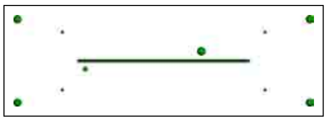
where $\sum_{j=1}^{l} \omega_j = 1$ and $\omega_j = \frac{1}{|\mathcal{C}_{\mathbf{p}}|} = \frac{1}{l}$.

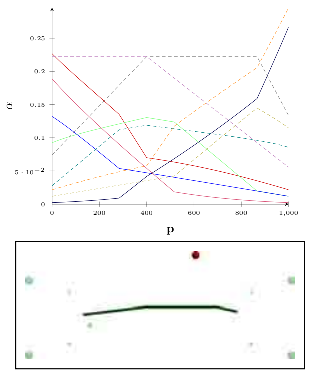
The coordinates in the second step are computed using one of the well-known existing systems of Generalised Barycentric Coordinates. The only requirement is that the selected system must coincide on the boundaries of the convexes in order to guarantee the continuity of the coordinates through them. Two examples of coordinate systems that fit our purpose are the Mean Value Coordinates and the Harmonic Coordinates. On the contrary, the Green Coordinates are not a good choice in this case since they do not match on the boundaries.

From the algorithm detailed above, we can see that the CA coordinates need to process the model multiple times. More precisely, it has to be processed l times, where l is the number of layers in the Catalog[4]. This point could be seen as a drawback because it causes the computation of the CA coordinates to be slower than in the case of the classical cage-based methods.

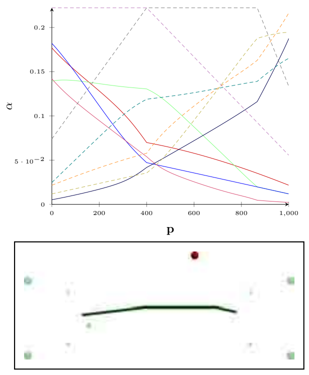
Another disadvantage of the CA coordinates is the C^0-continuity that they can present when the model intersects any face of the polyhedra of the layers, as can be noticed in Figures 3.23 and 3.24. This low level of continuity depends on the existing system of coordinates used to compute the CA coordinates on each layer. The examples in Figures 3.23 and 3.24 use the Mean Value Coordinates, which are C^0-continuous on the boundaries.

[4]Each layer contains the entire model, as stated before.

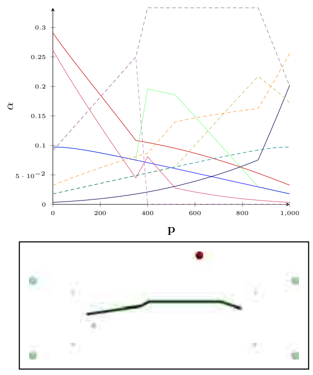




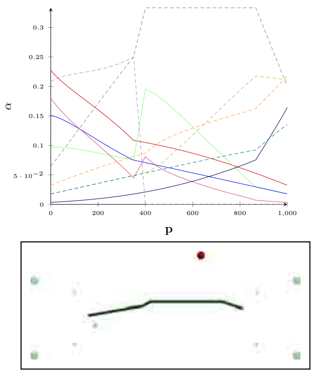
(a) Layers of independent partitions of tetrahedra.



(b) Layers of independent partitions of arbitrary polyhedra.



(c) Layers of nested partitions of tetrahedra.



(d) Layers of nested partitions of arbitrary polyhedra.

Figure 3.23: Examples of the CA coordinates computed over the different supporting layers described in this section. The coordinates have been computed by means of the Mean Value Coordinates, which makes the C^0-continuity of the functions arise. The top image shows the initial configuration of the control points around the model of the line.

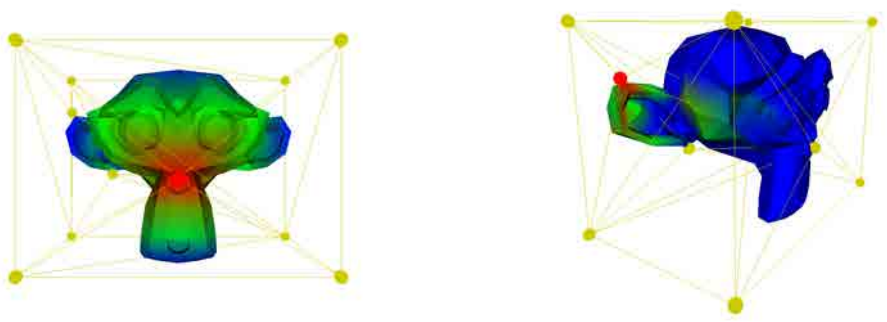


Figure 3.24: Example of the heat map of the CA coordinates for two different control points in the vicinity of the *Monkey* model. In this case, the coordinates have been computed by means of the Mean Value Coordinates. The area in red is the most influenced by the corresponding handle. In contrast, the area in blue remains constant when the control point is dragged. The complete Catalog, which consists of nested partitions of tetrahedra, is also shown. Notice the rough transition of the heat map from green to blue when the model crosses an *interior* face of the polyhedra of the Catalog. This situation is clearer in the image on the right.

Finally, Figures 3.25 and 3.26 show two different examples of deformations that use the CA coordinates. The Catalog that supports these deformations consists of nested partitions of tetrahedra and the final CA coordinates are computed by means of the Mean Value Coordinates.

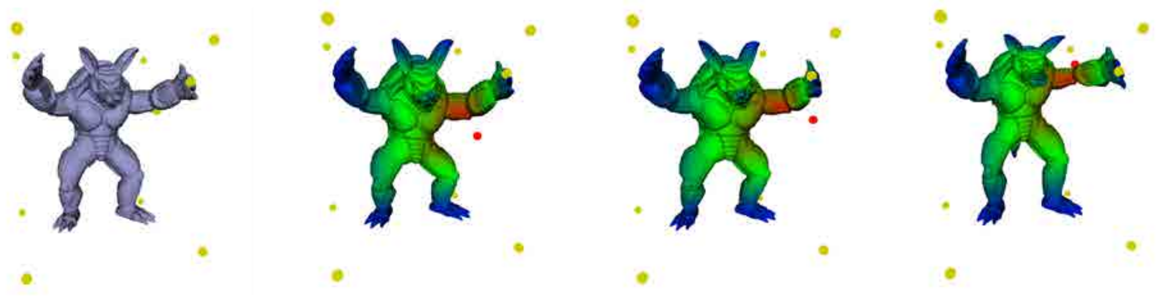


Figure 3.25: Example of a deformation of the *Armadillo* model by means of the CA coordinates. (a) shows the initial configuration of the *Armadillo* model surrounded by ten control points: eight handles located in the vertices of a wrapping cube and the remaining two located near the hand and the elbow of the *Armadillo*. The Catalog of Layers is built with nested partitions of tetrahedra. (b) shows the heat map and the result of displacing the control point close to the elbow. Notice that the control point located near the hand allows us to keep it in a more constant position while the control point in the elbow is displaced. The red area of the heat map is the most influenced by the control point, whereas the area in blue remains constant when the handle is dragged.

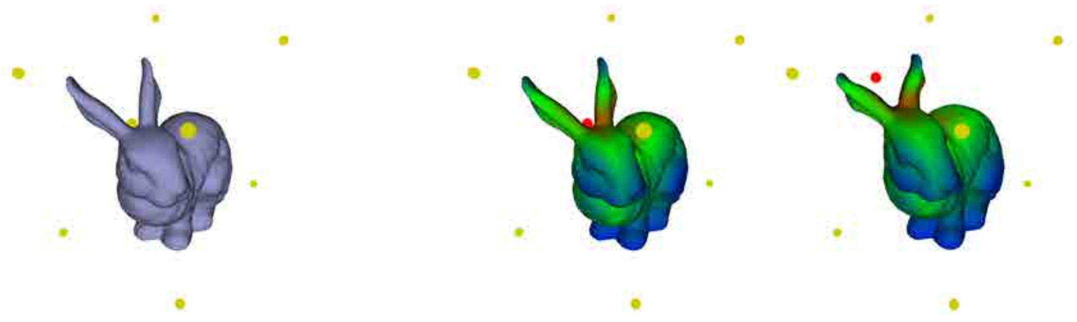


Figure 3.26: Example of a deformation of the *Bunny* model by means of the CA coordinates. (a) shows the initial configuration of the *Bunny* model surrounded by nine control points: eight handles located in the vertices of a wrapping cube and the remaining one located between the ears of the *Bunny*. The Catalog of Layers is built with nested partitions of tetrahedra. (b) shows the heat map and the result of displacing the control point close to the ears. The displacement of this handle produces an elongation of the head of the *Bunny* while the ears are shortened. The red area of the heat map is the most influenced by the control point, whereas the area in blue remains constant when the handle is dragged.

3.2.3 Conclusions

In this section we have studied some layered based methods which automatically perform a partition of the convex hull of the control points. This partition defines the supporting connectivity between the control points required to compute the coordinates. In addition, all the methods presented use the same supporting structure for all the points of the model, so the computed coordinates meet the continuity property which lack most of the techniques presented in Section 3.1.

However, the described layered based methods are not smooth or local enough, i.e. they present multiple local maxima and C^0-continuous functions. These disadvantages make them unintuitive for the user when he or she tries to apply them in a deformation process. Moreover, the performed deformations can present sharp features which lead to unpleasant results.

Faced with this fact, we conclude that we need to study how to increase the smoothness of the coordinate functions keeping in mind the continutity property. We also want a more local effect of the deformation. These new goals lead us to the next section, where we work on the n-Dimensional Space directly, being n the number of control points.

3.3 n-Dimensional Space Based Methods

In previous Sections 3.1 and 3.2 we have examined the cage-free problem in the three-Dimensional Space. However, all the presented approaches have important disadvantages in terms of continuity, smoothness, locality or efficiency. All these drawbacks are mostly caused by the automatically computed connectivity of the control points, used as the supporting structure to obtain a system of coordinates based on the well-known MVC. For this reason, we considered exploring new schemes to compute the coordinates that do not depend on the underlying connectivity of the handles.

In this section, we consider the problem from the n-Dimensional point of view, where n is the number of user-defined control points. In addition, we do not define any structure of these handles, but we try to directly solve systems of equations defined over them. Let us state our problem mathematically in the n-Dimensional space before we can proceed to define the studied approaches in detail.

We assume that the model to be deformed is completely contained in the convex hull of the control points $\mathcal{V}$. Then, given a model point $\mathbf{p}$ and the n control points $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$, we want to compute a set of coordinates $A = \{\alpha_1, \dots, \alpha_n\}$ such that

$$\mathbf{p} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i. \tag{3.14}$$

This equation can be developed as follows

$$\begin{aligned}\mathbf{0} &= \sum_{i=1}^{n} \alpha_1 (\mathbf{v}_i - \mathbf{p}) \\ &= \sum_{i=1}^{n} \alpha_i d_i \vec{\mathbf{q}}_i \\ &= \sum_{i=1}^{n} \beta_i \vec{\mathbf{q}}_i,\end{aligned} \tag{3.15}$$

where $\vec{\mathbf{q}}_i$ is the projection of the control point $\mathbf{v}_i$ over the unit sphere $\mathcal{S}_{\mathbf{p}}$ centered on $\mathbf{p}$, $\beta_i = \alpha_i d_i$ and $d_i = \|(\mathbf{v}_i - \mathbf{p})\|$. At this point, the three-dimensional System (3.15) can be expanded in the following three n-dimensional equations

$$0 = \sum_{i=1}^{n} \beta_i q_{i_x}, \qquad 0 = \sum_{i=1}^{n} \beta_i q_{i_y}, \qquad 0 = \sum_{i=1}^{n} \beta_i q_{i_z}.$$

Now, we define three n-dimensional vectors

$$\chi = \begin{pmatrix} q_{1_x} & \dots & q_{n_x} \end{pmatrix}, \quad y = \begin{pmatrix} q_{1_y} & \dots & q_{n_y} \end{pmatrix}, \quad z = \begin{pmatrix} q_{1_z} & \dots & q_{n_z} \end{pmatrix}. \tag{3.16}$$

Therefore, our goal is to find a vector $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 & \dots & \beta_n \end{pmatrix}$ such that

$$\boldsymbol{\beta} \cdot \chi = 0, \qquad \boldsymbol{\beta} \cdot y = 0, \qquad \boldsymbol{\beta} \cdot z = 0. \tag{3.17}$$

In other words, we want to compute a vector $\boldsymbol{\beta}$ perpendicular to the vectors χ, y and z. This vector $\boldsymbol{\beta}$ lies in the n-dimensional lineal subspace $V^\perp \subset \mathbb{R}^n$, where $V^\perp$ is defined as

$$V^\perp = \{\gamma : \gamma \cdot \chi = 0 \land \gamma \cdot y = 0 \land \gamma \cdot z = 0\}, \tag{3.18}$$

the subspace orthogonal to the control points. In addition, we require all the components of the vector $\boldsymbol{\beta}$ to be positive in order to get a well-defined system of coordinates. Then, we define $\mathbb{R}^n_+ \subset \mathbb{R}^n$ as the n-dimensional subset where all the components are positive and at least one of them is greater than 0

$$\mathbb{R}^n_+ = \{\gamma : \gamma_i \geq 0 \; \forall i \land \exists j : \gamma_j > 0, \;\; i, j \in [1, n]\}$$

or

$$\mathbb{R}^n_+ = \{\gamma : \gamma_i \geq 0 \; \forall i \in [1, n]\} \setminus \{\mathbf{0}\}. \tag{3.19}$$

If we put all the definitions together, we can affirm that the problem of computing the set of coordinates A is equivalent to find a vector $\boldsymbol{\beta} \in V^\perp \cap \mathbb{R}^n_+$. Therefore, we need to prove that $V^\perp \cap \mathbb{R}^n_+ \neq \emptyset$.

Proof that $V^\perp \cap \mathbb{R}^n_+ \neq \emptyset$. Given a particular point $\mathbf{p}$ of the model, we know that it is located inside the convex hull of the control points. The definition of the convex hull of a finite set of points states that any point inside it can be expressed as a convex combination of the vertices that define this convex hull. Thus, the interior of the convex hull of the control points $int(ch(\mathcal{V}))$ can be represented as

$$int(ch(\mathcal{V})) = \left\{ \mathbf{p} : \mathbf{p} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i, \; \alpha_i \geq 0 \; \forall \alpha_i \land \sum_{i=1}^{n} \alpha_i = 1 \right\}.$$

Now, we apply a translation over Equation (3.14) such that the new location of the point $\mathbf{p}$ is the origin $\mathbf{0}$ of the euclidean space

$$\begin{aligned} \mathbf{p} - \mathbf{p} &= \sum_{i=1}^{n} \alpha_i (\mathbf{v}_i - \mathbf{p}) \\ \mathbf{0} &= \sum_{i=1}^{n} \alpha_i (\mathbf{v}_i - \mathbf{p}). \end{aligned} \tag{3.20}$$

We know that $(\mathbf{v}_i - \mathbf{p}) = d_i \vec{\mathbf{q}}_i$, so we can substitute this expression into Equation (3.20)

$$\mathbf{0} = \sum_{i=1}^{n} (\alpha_i d_i) \vec{\mathbf{q}}_i.$$

We name $\beta_i = \alpha_i d_i$, so

$$\mathbf{0} = \sum_{i=1}^{n} \beta_i \vec{\mathbf{q}}_i.$$

Notice that the development above is exactly the same that we have done to get the System (3.15).

Finally, we know that $\alpha_i \geq 0$ and $d_i > 0$, by definition. Hence, $\beta_i \geq 0$.

At this stage, we have demonstrated that $\exists \boldsymbol{\beta} = (\beta_1, \dots, \beta_n)$ such that $\boldsymbol{\beta} \in \mathbb{R}^n_+$ and it is perpendicular to the vectors χ, y and z. Therefore, we can state that

$$\exists \boldsymbol{\beta} \in V^\perp \cap \mathbb{R}^n_+,$$

and, consequently,

$$V^\perp \cap \mathbb{R}^n_+ \neq \emptyset,$$

as we wanted to demonstrate. □

The following sections present different experiments to compute valid and correct vectors $\boldsymbol{\beta}$. These vectors $\boldsymbol{\beta}$ are the base to get a well-defined system of coordinates that relates the model points with the handles of the deformation.

3.3.1 Lagrange Minimisation Approach

The very first idea we had confronting the problem in the n-dimensional space was to define an optimisation over the coordinates vector $\boldsymbol{\beta}$. In other words, we defined some constraints that the coordinate functions had to meet. Thereby, we defined a target vector $U = (u_1, \dots, u_n) \in \mathbb{R}^n_+$. Then, the coordinates $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n)$ must be computed to satisfy the Equations in (3.17) while they were kept as close as possible to that target vector U.

The new technique, whose system was thought as a constrained optimisation problem, uses the Lagrange Minimisation methodology to find the coordinates β_i. Equation (3.21) expresses the described optimisation for this Lagrange Minimisation Approach (LMA).

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{Minimise}} \quad & f(\boldsymbol{\beta}) = \sum_{i=1}^{n} (\beta_i - u_i)^2 \\ \text{subject to} \quad & \boldsymbol{\beta} \cdot \chi = 0 \\ & \boldsymbol{\beta} \cdot y = 0 \\ & \boldsymbol{\beta} \cdot z = 0. \end{aligned} \tag{3.21}$$

Let us denote the constrained functions as

$$g_1(\boldsymbol{\beta}) = \boldsymbol{\beta} \cdot \chi, \qquad g_2(\boldsymbol{\beta}) = \boldsymbol{\beta} \cdot y, \qquad g_3(\boldsymbol{\beta}) = \boldsymbol{\beta} \cdot z.$$

Then, in order to use the Lagrange Minimisation method, the system defines three Lagrange multipliers $(\lambda_1, \lambda_2, \lambda_3)$ and the Lagrangian

$$L = f(\boldsymbol{\beta}) - \lambda_1 \cdot g_1(\boldsymbol{\beta}) - \lambda_2 \cdot g_2(\boldsymbol{\beta}) - \lambda_3 \cdot g_3(\boldsymbol{\beta}).$$

Finally, the coordinates β_i are obtained by the resolution of the System (3.22).

$$\begin{cases} \dfrac{\delta L}{\delta \beta_i} = & 0 \quad \forall i \in [1, n] \\ g_1(\boldsymbol{\beta}) = & 0 \\ g_2(\boldsymbol{\beta}) = & 0 \\ g_3(\boldsymbol{\beta}) = & 0 \end{cases} \tag{3.22}$$

Below, we expand the expression $\frac{\delta L}{\delta \beta_i} = 0$ and solve the System (3.22) analytically to find the LMA coordinates. Then,

$$\frac{\delta L}{\delta \beta_i} = 2(\beta_i - u_i) - \lambda_1 x_i - \lambda_2 y_i - \lambda_3 z_i = 0,$$

so

$$\beta_i = \frac{\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i + 2u_i}{2}. \tag{3.23}$$

Now, we can substitute the Expression (3.23) for β_i into $g_1(\boldsymbol{\beta})$, $g_2(\boldsymbol{\beta})$ and $g_3(\boldsymbol{\beta})$

$$\begin{aligned} g_1(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \beta_i \cdot u_i = \sum_{i=1}^{n} (\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i + 2u_i) \cdot x_i = 0 \\ g_2(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \beta_i \cdot u_i = \sum_{i=1}^{n} (\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i + 2u_i) \cdot y_i = 0 \\ g_3(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \beta_i \cdot u_i = \sum_{i=1}^{n} (\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i + 2u_i) \cdot z_i = 0. \end{aligned} \tag{3.24}$$

If we rewrite System (3.24) on

$$\begin{aligned} g_1(\boldsymbol{\beta}) &= \lambda_1 \sum_{i=1}^{n} (x_i \cdot x_i) + \lambda_2 \sum_{i=1}^{n} (x_i \cdot y_i) + \lambda_3 \sum_{i=1}^{n} (x_i \cdot z_i) + 2 \sum_{i=1}^{n} (u_i \cdot x_i) = 0 \\ g_2(\boldsymbol{\beta}) &= \lambda_1 \sum_{i=1}^{n} (x_i \cdot y_i) + \lambda_2 \sum_{i=1}^{n} (y_i \cdot y_i) + \lambda_3 \sum_{i=1}^{n} (y_i \cdot z_i) + 2 \sum_{i=1}^{n} (u_i \cdot y_i) = 0 \\ g_3(\boldsymbol{\beta}) &= \lambda_1 \sum_{i=1}^{n} (x_i \cdot z_i) + \lambda_2 \sum_{i=1}^{n} (y_i \cdot z_i) + \lambda_3 \sum_{i=1}^{n} (z_i \cdot z_i) + 2 \sum_{i=1}^{n} (u_i \cdot z_i) = 0, \end{aligned}$$

we can express it as the matrix system

$$\underbrace{\begin{pmatrix} \sum_{i=1}^{n}(x_i \cdot x_i) & \sum_{i=1}^{n}(x_i \cdot y_i) & \sum_{i=1}^{n}(x_i \cdot z_i) \\ \sum_{i=1}^{n}(x_i \cdot y_i) & \sum_{i=1}^{n}(y_i \cdot y_i) & \sum_{i=1}^{n}(y_i \cdot z_i) \\ \sum_{i=1}^{n}(x_i \cdot z_i) & \sum_{i=1}^{n}(y_i \cdot z_i) & \sum_{i=1}^{n}(z_i \cdot z_i) \end{pmatrix}}_{Q} \cdot \underbrace{\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix}}_{\lambda} = \underbrace{-2 \begin{pmatrix} \sum_{i=1}^{n}(u_i \cdot x_i) \\ \sum_{i=1}^{n}(u_i \cdot y_i) \\ \sum_{i=1}^{n}(u_i \cdot z_i) \end{pmatrix}}_{b.}$$

Finally, we can solve $\lambda = Q^{-1} \cdot b$ and, consequently, the coordinates β_i are also known when the λ values are substituted into Equation (3.23). The final coordinates are computed as

$$\mathbf{0} = \sum_{i=1}^{n} \beta_i \mathbf{q}_i = \sum_{i=1}^{n} \beta_i \frac{\mathbf{v}_i - \mathbf{p}}{\|\mathbf{v}_i - \mathbf{p}\|},$$

so

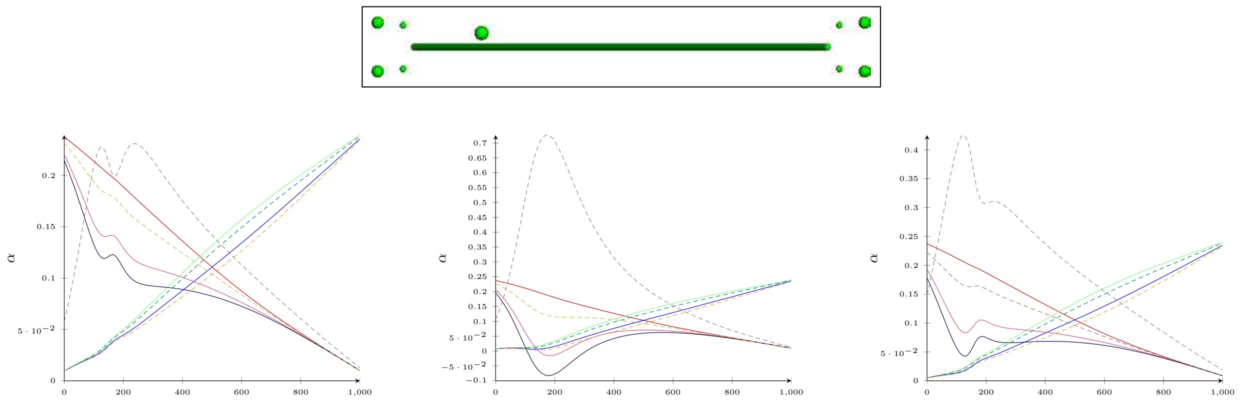
$$\alpha_i = \frac{\frac{\beta_i}{\|\mathbf{v}_i - \mathbf{p}\|}}{\sum_{j=1}^{n} \frac{\beta_j}{\|\mathbf{v}_j - \mathbf{p}\|}}.$$

The LMA coordinates can be solved for different values of the target vector $U = (u_1, \ldots, u_n)$ and we used three different vectors in our experiments:

- $u_i = 1$, $\forall i \in [1, n]$,

- $u_i = \frac{1}{\|\mathbf{v}_i - \mathbf{p}\|}$ and
- $u_i = 1 - \frac{\|\mathbf{v}_i - \mathbf{p}\|}{\max_{dist}(\mathcal{V}, \mathbf{p})}$, where $\max_{dist}(\mathcal{V}, \mathbf{p})$ is the distance between $\mathbf{p}$ and the farthest control point to it.

Although the LMA solution results in a smooth and continuous set of coordinates, the positivity of these is not ensured. Figure 3.27 shows three plots corresponding to our three target vectors U. All these plots are based on the same simple example, a line of points surrounded by nine control points. Notice that some of the coordinates present negative values in their functions.



(a) $U = (1, \ldots, 1)$ (b) $U = (\frac{1}{\|\mathbf{v}_1 - \mathbf{p}\|}, \ldots, \frac{1}{\|\mathbf{v}_n - \mathbf{p}\|})$ (c) $U = (1 - \frac{\|\mathbf{v}_1 - \mathbf{p}\|}{\max_{dist}(\mathcal{V}, \mathbf{p})}, \ldots, 1 - \frac{\|\mathbf{v}_n - \mathbf{p}\|}{\max_{dist}(\mathcal{V}, \mathbf{p})})$

Figure 3.27: Plots of the LMA coordinates that show the effect of different target vectors U over the coordinates. Observe that (b) presents some negative coordinate values.

Therefore, a second solution consisted in adding the positivity constraints to the optimisation system. Following what we thought the most intuitive and straightforward solution, we defined the new optimisation problem as

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{Minimise}} \quad & f(\boldsymbol{\beta}) = \sum_{i=1}^{n} (\beta_i - u_i)^2 \\ \text{subject to} \quad & \boldsymbol{\beta} \cdot \chi = 0 \\ & \boldsymbol{\beta} \cdot y = 0 \\ & \boldsymbol{\beta} \cdot z = 0 \\ & \beta_i \geq 0 \qquad\qquad \forall i \in [1, n]. \end{aligned} \tag{3.25}$$

Then, the new Lagrangian introduces the Kuhn-Tucker conditions

$$L = f(\boldsymbol{\beta}) - \lambda_1 \cdot g_1(\boldsymbol{\beta}) - \lambda_2 \cdot g_2(\boldsymbol{\beta}) - \lambda_3 \cdot g_3(\boldsymbol{\beta}) + \boldsymbol{\mu} \cdot \boldsymbol{\beta},$$

where $f(\boldsymbol{\beta})$, $g_1(\boldsymbol{\beta})$, $g_2(\boldsymbol{\beta})$ and $g_3(\boldsymbol{\beta})$ are the functions described above and $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ are the

Kuhn-Tucker constants. Then, the coordinates β_i are determined through the system

$$\begin{cases} \dfrac{\delta L}{\delta \beta_i} = & 0 \quad \forall i \in [1,n] \\ g_1(\boldsymbol{\beta}) = & 0 \\ g_2(\boldsymbol{\beta}) = & 0 \\ g_3(\boldsymbol{\beta}) = & 0 \\ \beta_i \geq & 0 \quad \forall i \in [1,n] \\ \mu_i\beta_i = & 0 \quad \forall i \in [1,n] \\ \mu_i \geq & 0 \quad \forall i \in [1,n]. \end{cases}$$

As done before, we also expand the expression $\frac{\delta L}{\delta \beta_i} = 0$ to analitycally isolate β_i

$$\beta_i = \frac{\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i - \mu_i + 2u_i}{2}, \tag{3.26}$$

and replace it into $g_1(\boldsymbol{\beta})$, $g_2(\boldsymbol{\beta})$, $g_3(\boldsymbol{\beta})$ and the $\mu_i\beta_i = 0$ conditions

$$\begin{aligned} g_1(\boldsymbol{\beta}) = \sum_{i=1}^{n} \beta_i \cdot u_i = \sum_{i=1}^{n} (\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i - \mu_i + 2u_i) \cdot x_i = &\ 0 \\ g_2(\boldsymbol{\beta}) = \sum_{i=1}^{n} \beta_i \cdot u_i = \sum_{i=1}^{n} (\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i - \mu_i + 2u_i) \cdot y_i = &\ 0 \\ g_3(\boldsymbol{\beta}) = \sum_{i=1}^{n} \beta_i \cdot u_i = \sum_{i=1}^{n} (\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i - \mu_i + 2u_i) \cdot z_i = &\ 0 \\ \mu_i(\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i - \mu_i + 2u_i) = &\ 0 \quad \forall i \in [1,n] \end{aligned} \tag{3.27}$$

The $\mu_i\beta_i = 0$ conditions are satisfied iff $\mu_i = 0$ or $\mu_i = \lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i + 2u_i$. Notice that the solution $\mu_i = \lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i + 2u_i$ involves $\beta_i = 0$ (by replacing the value μ_i into Equation (3.26)). Then, we can reduce the System (3.27) to a new system of three equations and three unknowns

$$\begin{aligned} g_1(\boldsymbol{\beta}) &= \sum_{i=1}^{n} (\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i - \mu_i + 2u_i) \cdot x_i = 0 \\ g_2(\boldsymbol{\beta}) &= \sum_{i=1}^{n} (\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i - \mu_i + 2u_i) \cdot y_i = 0 \\ g_3(\boldsymbol{\beta}) &= \sum_{i=1}^{n} (\lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i - \mu_i + 2u_i) \cdot z_i = 0, \end{aligned} \tag{3.28}$$

that satisfies the inequality conditions of the optimisation system depending on the two possible values of the Kuhn-Tucker constants μ_i. As far as there are n constants μ_i, there are 2^n possible results for the System (3.28) that need to be analysed in order to get a good solution. In what follows, we proceed to describe three different tests we tried to compute a well-defined set of coordinates out of the 2^n feasible possibilities.

Local Test

The Local Test studies the 2^n feasible results for the System (3.28) at each point $\mathbf{p}$ of the model, at first. Then, the solution that satisfies all the conditions in the optimisation System (3.25) and has the minimum number of $\beta_i = 0$ is chosen for that particular point $\mathbf{p}$.

The main problem of this methodology is its own locality. It tests the solutions at each point $\mathbf{p}$ locally and it does not have a global view of the behaviour of these solutions all over the model. Hence, discontinuities arise on the functions of the coordinates when, for two close points $\mathbf{p}_1$ and $\mathbf{p}_2$, the chosen combination for the μ_i values changes. See Figure 3.28, which shows discontinuous coordinates for the basic example of the line of points surrounded by nine control points.

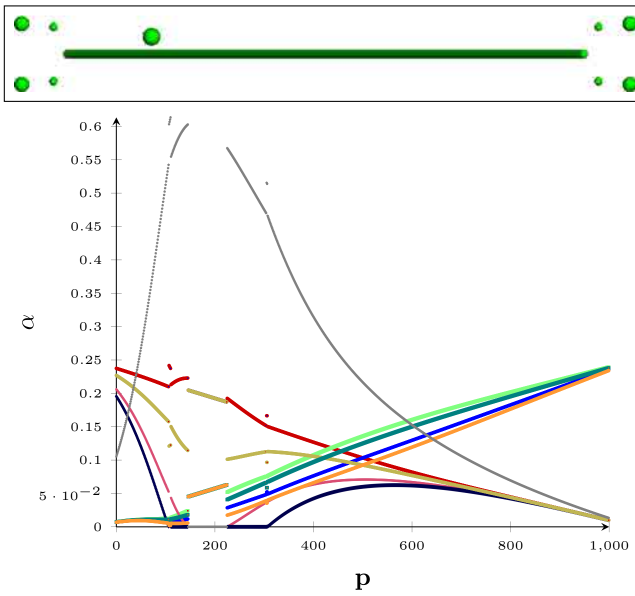


Figure 3.28: The Local Test methodology produces discontinuities in the coordinates. In this example, we have used the target vector $U = (\frac{1}{\|\mathbf{v}_1 - \mathbf{p}\|}, \ldots, \frac{1}{\|\mathbf{v}_n - \mathbf{p}\|})$.

On the other hand, this methodology is a kind of brute force solution because it needs to do a lot of tests. Therefore, a lot of time is spent to compute the final set of coordinates.

Global Test

It was reasonable to think that, if the problem of the above methodology was caused by the locality of the tests, a global analysis of the optimisation problem should give better results. For this reason, the Global Test computes the 2^n feasible solutions for System (3.28) for all the points $\mathbf{p}$ of the model. Then, it chooses the combination of the μ_i values that satisfies all the conditions of the optimisation System (3.25) for all the points at the same time and has the minimum number of $\beta_i = 0$. However, this methodology also has a disadvantage: if the selected combination generates any $\beta_i = 0$, the weight β_i and its corresponding final coordinate α_i are equal to 0 for the entire model. This means that the control point $\mathbf{v}_i$ will not affect the model and no modification of its location will be translated into a deformation of the object. In addition, the Global Test is as inefficient as the Local Test since it also computes all the 2^n feasible possibilities at each model point $\mathbf{p}$.

Iterative Test

The two methodologies above are not valid to get a well-defined set of coordinates due to the discontinuities or the null values of the β_i weights. Thus, we studied a further method to choose the right combination of the μ_i values which performed multiple tests iteratively.

First, the Iterative Test solves the Lagrange Minimisation assuming all $\mu_i = 0$. This could be considered an optimal solution if no $\beta_i < 0$ were generated. Notice also that the solution where all $\mu_i = 0$ is equivalent to solve the optimisation system in (3.21). Then, if there exists any $\beta_i < 0$, the corresponding μ_i value is replaced by $\mu_i = \lambda_1 x_i + \lambda_2 y_i + \lambda_3 z_i + 2u_i$ and the method solves the system again in a new iteration. It repeats this process until all the values β_i are zero or positive.

This solution gives a continuous set of coordinates. However, when one or more weights β_i are 0, the continuity is only C^0. Figure 3.29 shows this problem while Figure 3.30 presents some deformations of the *Armadillo* model with unpleasant shapes caused by the C^0-continuity regions.

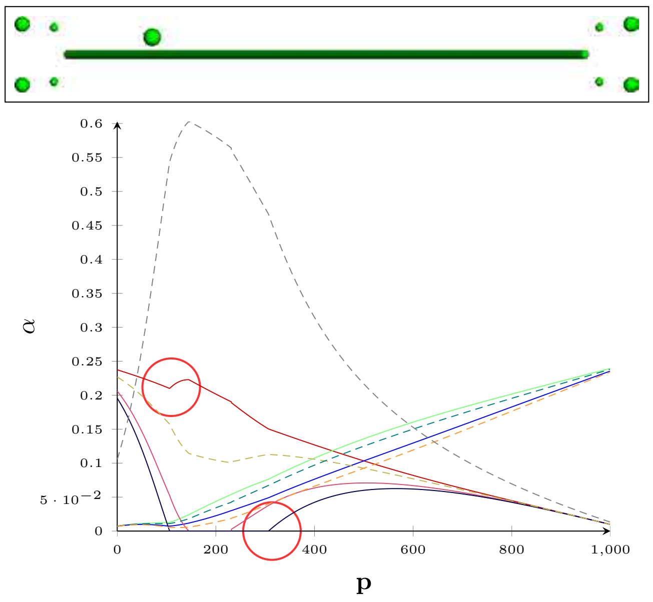


Figure 3.29: The Iterative Test methodology causes C^0-continuity in the areas where any of the weights β_i are 0. The red circles highlight this singularity in two different points. This example uses the target vector $U = (\frac{1}{\|\mathbf{v}_1 - \mathbf{p}\|}, \ldots, \frac{1}{\|\mathbf{v}_n - \mathbf{p}\|})$.

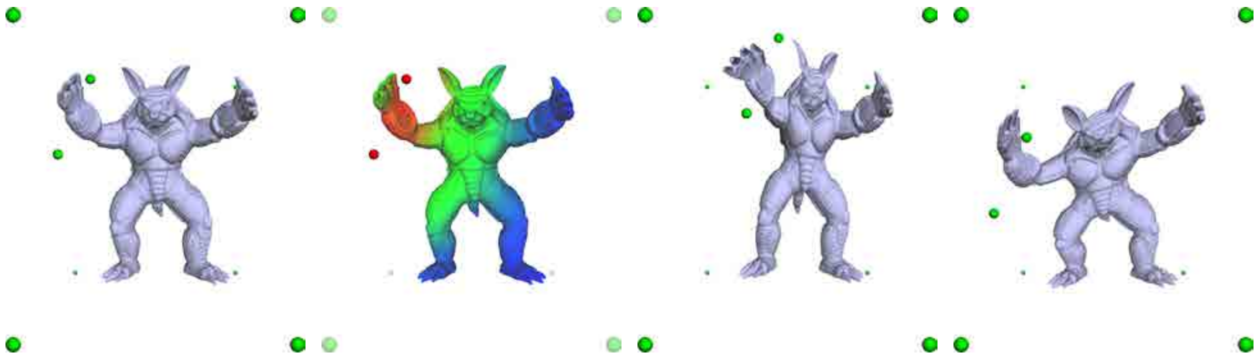


Figure 3.30: Deformation and heat map produced by the Iterative Test methodology.

None of the three analysed tests give good enough results. However, the Iterative Test led us to think of yet another, easy and iterative method. We called it the Iterative Lagrange Minimisation Approach, as explained below.

Iterative Lagrange Minimisation Approach

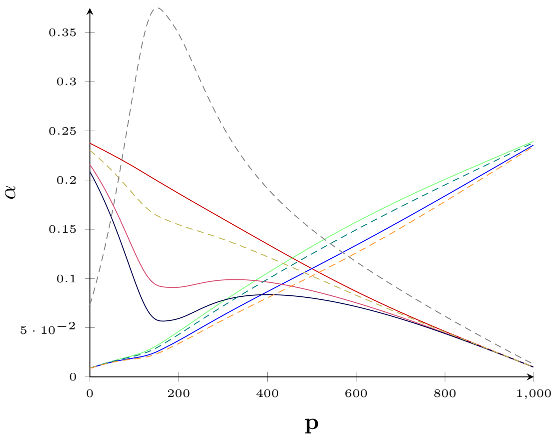
The Iterative Lagrange Minimisation Approach (ILMA) solves the optimisation system

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{Minimise}} \quad & f(\boldsymbol{\beta}) = \sum_{i=1}^{n} (\beta_i - u_i)^2 \\ \text{subject to} \quad & \boldsymbol{\beta} \cdot \chi = 0 \\ & \boldsymbol{\beta} \cdot y = 0 \\ & \boldsymbol{\beta} \cdot z = 0, \end{aligned} \tag{3.29}$$

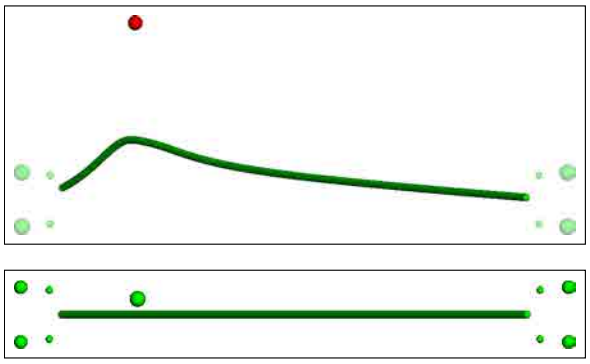
which does not take into account the constraints over the positivity of the β_i weights. Once the System (3.29) is solved for all the points $\mathbf{p}$ of the model, the ILMA looks for the absolute minimum of the weights β_i. That is, the minimum value β_i among all the model points $\boldsymbol{\beta}$ coordinates. We call this value β_{min}. Then, if β_{min} is negative, the method updates the target vector U such that

$$u'_i = u_i + \beta_{min} \quad \forall i \in [1, n]$$

and solves the Lagrange Minimisation again, using the updated vector U. This process is repeated until all the coordinates β_i are positive or zero. Figure 3.31 shows the plot of the coordinates for the classic line model surrounded by nine control points. Notice how the negative coordinates shown in Figure 3.27(b) have now disappeared due to the consecutive minimisation iterations.



(a)



(b)

Figure 3.31: Figure (a) shows the ILMA coordinate functions corresponding to the initial configuration of a line surrounded by nine control points. The bottom image in (b) presents the initial configuration while the top image shows a simple deformation produced by the translation of the handle in red. The coordinates have been computed by means of the targed vector $U = (\frac{1}{\|\mathbf{v}_1 - \mathbf{p}\|}, \ldots, \frac{1}{\|\mathbf{v}_n - \mathbf{p}\|})$.

Unfortunately, the convergence of the iterative method used to compute the ILMA coordinates is not always guaranteed. Consequently, the ILMA is not suitable enough and, thus, we kept researching new techniques to build a well-defined system of coordinates.

3.3.2 Wire Approach

Once the experiments with the Lagrange Minimisation Approaches were finished, we proceeded with our next attempt to compute a well-defined system of coordinates. The main idea behind this new method was to build a wire-like structure over the linear subspace $V^{\perp}$ and to intersect it with the $\mathbb{R}^n_+$ region, so we called it the Wire Approach (WA).

In order to build the wire structure, the method selects a set of points $\mathcal{P} = \{p \in \mathbb{R}^n\}$ and projects them over $V^{\perp}$. Then, the WA connects the projected points $\bar{p} \in V^{\perp}$ and intersects the newly created segments with the $\mathbb{R}^n_+$ region. This intersection defines the portion of the wire structure laying on $\mathbb{R}^n_+$. Hence, we can compute its centroid to get the final coordinates.

There are different ways to choose the points $p \in \mathcal{P}$ and to connect them. In the following paragraphs we analyse the different options used in our experiments and discuss the derived coordinates for each of them.

Set of points $\mathcal{P}$

We defined the set of points $\mathcal{P}$ in three different ways: $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$. The set $\mathcal{P}_1$ includes n points, each of them containing a single one in their components while the rest are zero. That is

$$\mathcal{P}_1 = \{p_i : p_{i,i} = 1 \wedge p_{i,j} = 0,\ \forall j \neq i\}, \quad |\mathcal{P}_1| = n.$$

The set $\mathcal{P}_2$ is very similar to $\mathcal{P}_1$. It also includes n points, but this time these points contain a single zero in their components while the rest are one. That is

$$\mathcal{P}_2 = \{p_i : p_{i,i} = 0 \wedge p_{i,j} = 1,\ \forall j \neq i\}, \quad |\mathcal{P}_2| = n.$$

Finally, the set $\mathcal{P}_3$ includes $2n$ points, combining points in $\mathbb{R}^n_+$ and points in $\mathbb{R}^n_-$. It is composed by the same points defining $\mathcal{P}_1$ and also by their opposite. That is

$$\mathcal{P}_3 = \{p_i, p_{n+i} : (p_{i,i} = 1 \wedge p_{i,j} = 0) \wedge (p_{n+i,i} = -1 \wedge p_{n+i,j} = 0),\ \forall j \neq i\}, \quad |\mathcal{P}_3| = 2n.$$

Regardless of the definition of the set $\mathcal{P}$ as $\mathcal{P}_1$, $\mathcal{P}_2$ or $\mathcal{P}_3$, the projection of its points p over $V^{\perp}$ is always performed by means of the projection matrix $P_{V^{\perp}}$ defined in Annex A.

Connection of the points

Once the set $\mathcal{P}$ was established, we also analysed two basic ways to connect its points. Firstly, we defined a closed cycle passing through all the points $p \in \mathcal{P}$. If $\mathcal{P}$ is defined as $\mathcal{P}_1$ or $\mathcal{P}_2$, then, the cyclic polyline $\mathcal{C}_1$ is equivalent to the sorted list

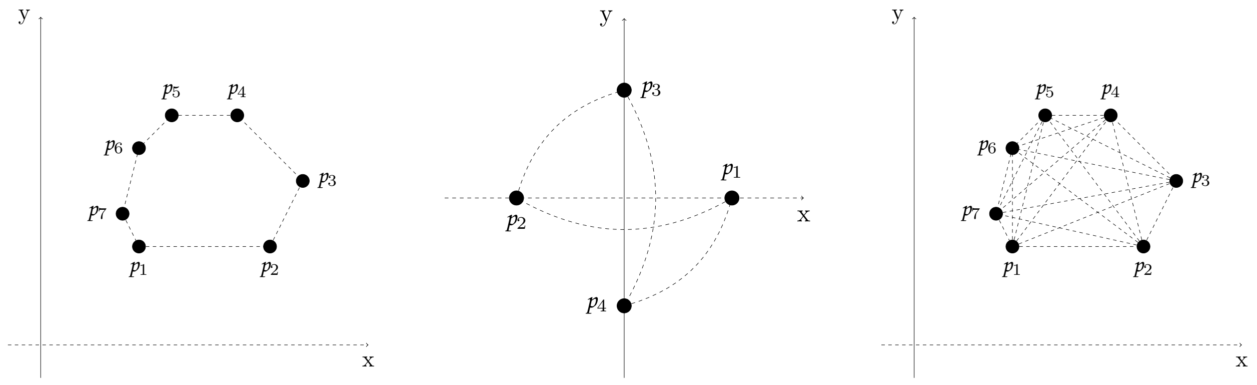
$$\mathcal{C}_1 = [p_1, p_2, \ldots, p_n, p_1].$$

On the other hand, if $\mathcal{P}$ is defined as $\mathcal{P}_3$, then, the cyclic polyline $\mathcal{C}_2$ is the sorted list

$$\mathcal{C}_2 = [p_1, p_{n+1}, p_2, p_{n+2}, \ldots, p_i, p_{n+i}, \ldots, p_n, p_{2n}, p_1].$$

As a second choice, we defined a complete graph to connect the points $p \in \mathcal{P}$. In other words, the connectivity $\mathcal{C}_3$ consists of connecting each point p to all the others. Figure 3.32 shows an example of $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ in the 2D space.

Finally, polylines $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ are projected to $V^{\perp}$.



(a) Cyclic connectivity $\mathcal{C}_1$ (b) Cyclic connectivity $\mathcal{C}_2$ (c) Complete graph $\mathcal{C}_3$

Figure 3.32: Examples of the connectivity between the points $p \in \mathcal{P}$. (a) shows an example of C_1 connectivity, (b) presents an example of $\mathcal{C}_2$ cycle and, finally, (c) illustrates an example of $\mathcal{C}_3$ graph.

Centroid

The WA method computes the centroid of the intersected portion of the wire structure. Thus, for every wire segment l_i laying in $\mathbb{R}^n_+$ with length ℓ_i, it computes its middle point q_i. Then, and assuming that there are m segments over $\mathbb{R}^n_+$, the centroid $\boldsymbol{\beta}$ of the wire is

$$\boldsymbol{\beta} = \frac{\sum_{i=1}^{m} q_i \ell_i}{m}.$$

Once $\boldsymbol{\beta}$ is computed, the technique obtains the final coordinates as

$$\mathbf{0} = \sum_{i=1}^{n} \beta_i \mathbf{q}_i = \sum_{i=1}^{n} \beta_i \frac{\mathbf{v}_i - \mathbf{p}}{\|\mathbf{v}_i - \mathbf{p}\|},$$

so

$$\alpha_i = \frac{\frac{\beta_i}{\|\mathbf{v}_i - \mathbf{p}\|}}{\sum_{j=1}^{n} \frac{\beta_j}{\|\mathbf{v}_j - \mathbf{p}\|}}.$$

Figure 3.33 shows the intersection between the wire structure and the $\mathbb{R}^n_+$ region for two close points $\mathbf{p}_1$ and $\mathbf{p}_2$.

Although the computation of the centroid explained above results in a set of well-defined coordinates, the intersection of the wire structure with the $\mathbb{R}^n_+$ region does not always exist. That is, there are wire structures that do not intersect $\mathbb{R}^n_+$. Consequently, there are parts of the model, or even the entire model, that can not be defined with respect to the control points. These parts can never be deformed, so the WA coordinates are not suitable to perform 3D deformations.

The existence of null intersections between the wire structure and the $\mathbb{R}^n_+$ is caused by the fact that the probability that this wire, with only n or $2n$ vertices, is projected on the $\mathbb{R}^n_+$ region rapidly decreases as n, the number of control points, increases. Note that $\mathbb{R}^n$ has 2^n regions and we are only interested in one of them. The studied set of points $\mathcal{P}$ does not sample sufficiently well the $\mathbb{R}^n$ space, so the WA is not always able to find the $\mathbb{R}^n_+$ region. For this reason, we were forced to continue researching new methodologies to obtain a well-defined system of coordinates.

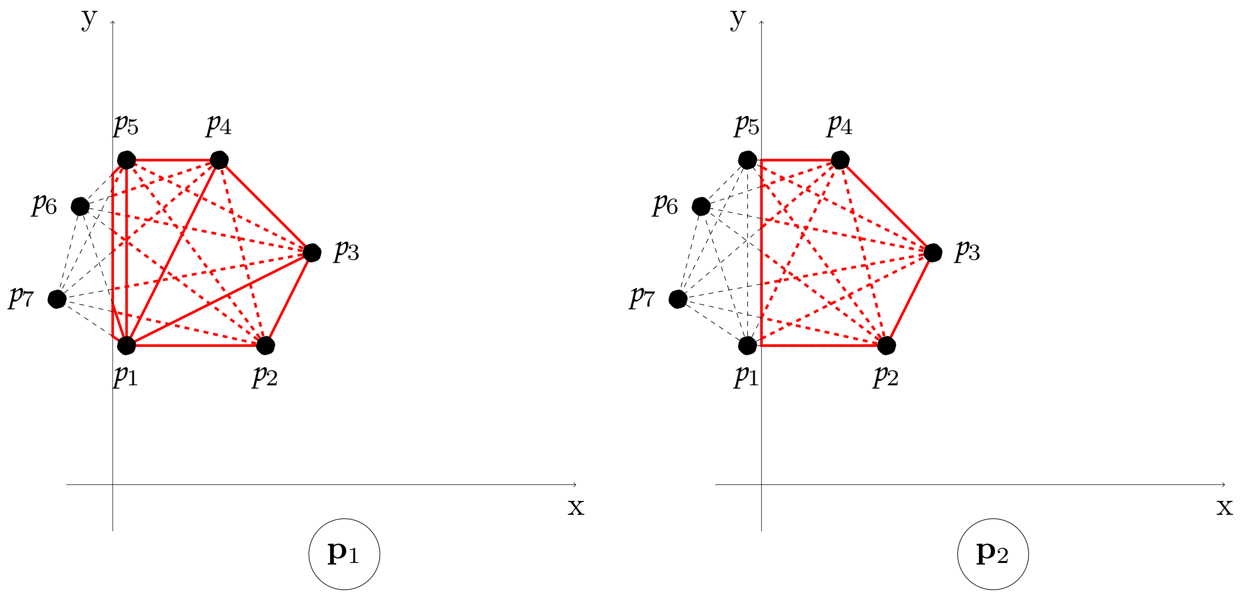


Figure 3.33: Example of the intersection between two wire structures and the $\mathbb{R}^n_+$ region. The red segments take part in the computation of the centroid while the dashed segments are ignored.

3.3.3 Determinants Approach

In our next attempt, we studied all the possible configurations of four control points that formed a non-degenerated tetrahedron. Notice that we could choose the four control points of a configuration in $\frac{n!}{4!(n-4)!}$ different ways, where n is the number of handles. However, given a particular model point $\mathbf{p}$, we computed its coordinates by taking into account only those tetrahedra containing it. The resulting coordinates were a combination of the Barycentric Coordinates of $\mathbf{p}$ with respect to each of the considered tetrahedra.

In order to discern which tetrahedra collaborated in the computation of the coordinates of $\mathbf{p}$, we analysed the determinants of the control points of each of them, choosing three at a time. Those determinants would also be used to compute intermediate Spherical Barycentric Coordinates (SBC) of $\mathbf{p}$, so we called the new technique the Determinants Approach (DA). Following these lines we explain in detail how DA exactly works.

Given a point $\mathbf{p}$ of the model, the main goal of the DA is to find the m tetrahedra containing it and to build a set $B_{\mathbf{p}} = \{\boldsymbol{\beta}^1, \dots, \boldsymbol{\beta}^m\}$ such that $B_{\mathbf{p}} \subset V^{\perp} \cap \mathbb{R}^n_+$. That is

$$\forall \boldsymbol{\beta}^j \in B_{\mathbf{p}} \quad : \quad \boldsymbol{\beta}^j \cdot \chi = 0 \quad \wedge \quad \boldsymbol{\beta}^j \cdot y = 0 \quad \wedge \quad \boldsymbol{\beta}^j \cdot z = 0 \quad \wedge \quad \boldsymbol{\beta}^j \in \mathbb{R}^n_+.$$

Each $\boldsymbol{\beta}^j$ vector will contain the SBC of $\mathbf{p}$ with respect to the j-th considered tetrahedron.

First, the method defines the matrix M as

$$M = \begin{pmatrix} \chi \\ y \\ z \end{pmatrix} = \begin{pmatrix} q_{1x} & q_{2x} & \dots & q_{nx} \\ q_{1y} & q_{2y} & \dots & q_{ny} \\ q_{1z} & q_{2z} & \dots & q_{nz} \end{pmatrix}$$

where each column c_i corresponds to the projected control point $\widehat{\mathbf{v}}_i$ on the unit sphere $\mathcal{S}_\mathbf{p}$ of $\mathbf{p}$. Remember from Section 3.1 that

$$\vec{\mathbf{q}}_i = \widehat{\mathbf{v}}_i - \mathbf{p} = \frac{\mathbf{v}_i - \mathbf{p}}{\|\mathbf{v}_i - \mathbf{p}\|}.$$

Then, given a particular tetrahedron $\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_l\}$, the method discards or includes it into the set of tetrahedra affecting $\mathbf{p}$ depending on the four determinats defined by its vertices, chosen in threes. The technique uses the indices $\{i, j, k, l\}$ of the control points of the tetrahedron, which we will call configuration from now on, to set up the list $L = [c_i, c_j, c_k, c_l, c_i, c_j]$ with the corresponding columns of M. It also computes four 3×3 matrices taking 3 consecutive columns in L

$$M_1 = \begin{pmatrix} c_i & c_j & c_k \end{pmatrix}, \quad M_2 = \begin{pmatrix} c_j & c_k & c_l \end{pmatrix}, \quad M_3 = \begin{pmatrix} c_k & c_l & c_i \end{pmatrix}, \quad M_4 = \begin{pmatrix} c_l & c_i & c_j \end{pmatrix}.$$

Finally, iff the configuration $\{i, j, k, l\}$ satisfies that the determinants of M_1, M_2, M_3 and M_4 alternate their signs, that is

$$M_1 \geq 0 \wedge M_3 \geq 0 \text{ and } M_2 \leq 0 \wedge M_4 \leq 0$$

or

$$M_1 \leq 0 \wedge M_3 \leq 0 \text{ and } M_2 \geq 0 \wedge M_4 \geq 0,$$

the tetrahedron $\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_l\}$ is used to compute the coordinates of $\mathbf{p}$. It is also necessary the point $\mathbf{p}$ to lie inside the tetrahedron defined by the control points $\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_l\}$.

If the tetrahedron $\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_l\}$ is correctly chosen, then, its corresponding vector $\boldsymbol{\beta}^j$ is

$$\boldsymbol{\beta}^j = \left(0, \ldots, 0, \overbrace{\det(M_2)}^{\beta_i^j}, 0, \ldots, 0, \overbrace{-\det(M_3)}^{\beta_j^j}, 0, \ldots, 0, \overbrace{\det(M_4)}^{\beta_k^j}, 0, \ldots, 0, \overbrace{-\det(M_1)}^{\beta_l^j}, 0, \ldots, 0 \right)$$

assuming $M_1 \leq 0 \wedge M_3 \leq 0$ and $M_2 \geq 0 \wedge M_4 \geq 0$ without loss of generality. Notice that $\frac{\boldsymbol{\beta}^j}{vol(\{\vec{\mathbf{q}}_i, \vec{\mathbf{q}}_j, \vec{\mathbf{q}}_k, \vec{\mathbf{q}}_l\})}$, where $vol(\{\vec{\mathbf{q}}_i, \vec{\mathbf{q}}_j, \vec{\mathbf{q}}_k, \vec{\mathbf{q}}_l\})$ is the volume of the tetrahedron $\{\widehat{\mathbf{v}}_i, \widehat{\mathbf{v}}_j, \widehat{\mathbf{v}}_k, \widehat{\mathbf{v}}_l\}$, corresponds to the SBC of $\mathbf{p}$ with respect to this tetrahedron $\{\widehat{\mathbf{v}}_i, \widehat{\mathbf{v}}_j, \widehat{\mathbf{v}}_k, \widehat{\mathbf{v}}_l\}$.

Assuming that the configuration $\{i, j, k, l\}$ is correct for the particular model point $\mathbf{p}$, we prove that $\boldsymbol{\beta}^j \cdot \chi = 0$ and $\boldsymbol{\beta}^j \cdot y = 0$ and $\boldsymbol{\beta}^j \cdot z = 0$ below.

Proof of $\boldsymbol{\beta} j \cdot \chi = 0$ *and* $\boldsymbol{\beta}^j \cdot y = 0$ *and* $\boldsymbol{\beta}^j \cdot z = 0$. First of all, let us refresh the nomenclature we need to develop this proof. The column vector c_i corresponds to the control point $\vec{\mathbf{q}}_i$

$$c_i = \vec{\mathbf{q}}_i^\top = \begin{pmatrix} q_{ix} \\ q_{iy} \\ q_{iz} \end{pmatrix},$$

as well as the component i of the vectors χ, y and z

$$\vec{\mathbf{q}}_i = (q_{ix}, q_{iy}, q_{iz}) = (\chi_i, y_i, z_i). \tag{3.30}$$

Now, we expand the determinants of the matrices M_1, M_2, M_3 and M_4 related to the list L:

$$\begin{aligned}
\beta_l^j &= -\det(M_1) = -\begin{vmatrix} c_i & c_j & c_k \end{vmatrix} = -\begin{vmatrix} q_{ix} & q_{jx} & q_{kx} \\ q_{iy} & q_{jy} & q_{ky} \\ q_{iz} & q_{jz} & q_{kz} \end{vmatrix} \\
&= -q_{ix}q_{jy}q_{kz} - q_{jx}q_{ky}q_{iz} - q_{kx}q_{iy}q_{jz} + q_{kx}q_{jy}q_{iz} + q_{jx}q_{iy}q_{kz} + q_{ix}q_{ky}q_{jz} \\
\beta_i^j &= \det(M_2) = \begin{vmatrix} c_j & c_k & c_l \end{vmatrix} = \begin{vmatrix} q_{jx} & q_{kx} & q_{lx} \\ q_{jy} & q_{ky} & q_{ly} \\ q_{jz} & q_{kz} & q_{lz} \end{vmatrix} \\
&= q_{jx}q_{ky}q_{lz} + q_{kx}q_{ly}q_{jz} + q_{lx}q_{jy}q_{kz} - q_{lx}q_{ky}q_{jz} - q_{kx}q_{jy}q_{lz} - q_{jx}q_{ly}q_{kz} \\
\beta_j^j &= -\det(M_3) = -\begin{vmatrix} c_k & c_l & c_i \end{vmatrix} = -\begin{vmatrix} q_{kx} & q_{lx} & q_{ix} \\ q_{ky} & q_{ly} & q_{iy} \\ q_{kz} & q_{lz} & q_{iz} \end{vmatrix} \\
&= -q_{kx}q_{ly}q_{iz} - q_{lx}q_{iy}q_{kz} - q_{ix}q_{ky}q_{lz} + q_{ix}q_{ly}q_{kz} + q_{lx}q_{ky}q_{iz} + q_{kx}q_{iy}q_{lz} \\
\beta_k^j &= \det(M_4) = \begin{vmatrix} c_l & c_i & c_j \end{vmatrix} = \begin{vmatrix} q_{lx} & q_{ix} & q_{jx} \\ q_{ly} & q_{iy} & q_{jy} \\ q_{lz} & q_{iz} & q_{jz} \end{vmatrix} \\
&= q_{lx}q_{iy}q_{jz} + q_{ix}q_{jy}q_{lz} + q_{jx}q_{ly}q_{iz} - q_{jx}q_{iy}q_{lz} - q_{ix}q_{ly}q_{jz} - q_{lx}q_{jy}q_{iz}.
\end{aligned}$$

Then, we expand the scalar products

$$\begin{aligned}
\boldsymbol{\beta}^j \cdot x &= \beta_i^j x_i + \beta_j^j x_j + \beta_k^j x_k + \beta_l^j x_l \\
&= \det(M_2)x_i - \det(M_3)x_j + \det(M_4)x_k - \det(M_1)x_l \\
\boldsymbol{\beta}^j \cdot y &= \beta_i^j y_i + \beta_j^j y_j + \beta_k^j y_k + \beta_l^j y_l \\
&= \det(M_2)y_i - \det(M_3)y_j + \det(M_4)y_k - \det(M_1)y_l \\
\boldsymbol{\beta}^j \cdot z &= \beta_i^j z_i + \beta_j^j z_j + \beta_k^j z_k + \beta_l^j z_l \\
&= \det(M_2)z_i - \det(M_3)z_j + \det(M_4)z_k - \det(M_1)z_l.
\end{aligned}$$

And,

$$\det(M_2)\cdot\vec{\mathbf{q}}_i^\top = \begin{pmatrix} q_{ix}q_{jx}q_{ky}q_{lz} + q_{ix}q_{jz}q_{kx}q_{ly} + q_{ix}q_{jy}q_{kz}q_{lx} - q_{ix}q_{jz}q_{ky}q_{lx} - q_{ix}q_{jy}q_{kx}q_{lz} - q_{ix}q_{jx}q_{kz}q_{ly} \\ q_{iy}q_{jx}q_{ky}q_{lz} + q_{iy}q_{jz}q_{kx}q_{ly} + q_{iy}q_{jy}q_{kz}q_{lx} - q_{iy}q_{jz}q_{ky}q_{lx} - q_{iy}q_{jy}q_{kx}q_{lz} - q_{iy}q_{jx}q_{kz}q_{ly} \\ q_{iz}q_{jx}q_{ky}q_{lz} + q_{iz}q_{jz}q_{kx}q_{ly} + q_{iz}q_{jy}q_{kz}q_{lx} - q_{iz}q_{jz}q_{ky}q_{lx} - q_{iz}q_{jy}q_{kx}q_{lz} - q_{iz}q_{jx}q_{kz}q_{ly} \end{pmatrix}$$

$$-\det(M_3)\cdot\vec{\mathbf{q}}_j^\top = \begin{pmatrix} q_{ix}q_{jx}q_{kz}q_{ly} + q_{iz}q_{jx}q_{ky}q_{lx} + q_{iy}q_{jx}q_{kx}q_{lz} - q_{iz}q_{jx}q_{kx}q_{ly} - q_{iy}q_{jx}q_{kz}q_{lx} - q_{ix}q_{jx}q_{ky}q_{lz} \\ q_{ix}q_{jy}q_{kz}q_{ly} + q_{iz}q_{jy}q_{ky}q_{lx} + q_{iy}q_{jy}q_{kx}q_{lz} - q_{iz}q_{jy}q_{kx}q_{ly} - q_{iy}q_{jy}q_{kz}q_{lx} - q_{ix}q_{jy}q_{ky}q_{lz} \\ q_{ix}q_{jz}q_{kz}q_{ly} + q_{iz}q_{jz}q_{ky}q_{lx} + q_{iy}q_{jz}q_{kx}q_{lz} - q_{iz}q_{jz}q_{kx}q_{ly} - q_{iy}q_{jz}q_{kz}q_{lx} - q_{ix}q_{jz}q_{ky}q_{lz} \end{pmatrix}$$

$$\det(M_4)\cdot\vec{\mathbf{q}}_k^\top = \begin{pmatrix} q_{iy}q_{jz}q_{kx}q_{lx} + q_{ix}q_{jy}q_{kx}q_{lz} + q_{iz}q_{jx}q_{kx}q_{ly} - q_{iy}q_{jx}q_{kx}q_{lz} - q_{ix}q_{jz}q_{kx}q_{ly} - q_{iz}q_{jy}q_{kx}q_{lx} \\ q_{iy}q_{jz}q_{ky}q_{lx} + q_{ix}q_{jy}q_{ky}q_{lz} + q_{iz}q_{jx}q_{ky}q_{ly} - q_{iy}q_{jx}q_{ky}q_{lz} - q_{ix}q_{jz}q_{ky}q_{ly} - q_{iz}q_{jy}q_{ky}q_{lx} \\ q_{iy}q_{jz}q_{kz}q_{lx} + q_{ix}q_{jy}q_{kz}q_{lz} + q_{iz}q_{jx}q_{kz}q_{ly} - q_{iy}q_{jx}q_{kz}q_{lz} - q_{ix}q_{jz}q_{kz}q_{ly} - q_{iz}q_{jy}q_{kz}q_{lx} \end{pmatrix}$$

$$-\det(M_1)\cdot\vec{\mathbf{q}}_l^\top = \begin{pmatrix} q_{iz}q_{jy}q_{kx}q_{lx} + q_{iy}q_{jx}q_{kz}q_{lx} + q_{ix}q_{jz}q_{ky}q_{lx} - q_{ix}q_{jy}q_{kz}q_{lx} - q_{iz}q_{jx}q_{ky}q_{lx} - q_{iy}q_{jz}q_{kx}q_{lx} \\ q_{iz}q_{jy}q_{kx}q_{ly} + q_{iy}q_{jx}q_{kz}q_{ly} + q_{ix}q_{jz}q_{ky}q_{ly} - q_{ix}q_{jy}q_{kz}q_{ly} - q_{iz}q_{jx}q_{ky}q_{ly} - q_{iy}q_{jz}q_{kx}q_{ly} \\ q_{iz}q_{jy}q_{kx}q_{lz} + q_{iy}q_{jx}q_{kz}q_{lz} + q_{ix}q_{jz}q_{ky}q_{lz} - q_{ix}q_{jy}q_{kz}q_{lz} - q_{iz}q_{jx}q_{ky}q_{lz} - q_{iy}q_{jz}q_{kx}q_{lz} \end{pmatrix}.$$

Therefore,

$$\begin{aligned}
\boldsymbol{\beta}^j \cdot \mathscr{x} &= q_{ix}q_{jx}q_{ky}q_{lz} + q_{ix}q_{jz}q_{kx}q_{ly} + q_{ix}q_{jy}q_{kz}q_{lx} - q_{ix}q_{jz}q_{ky}q_{lx} - q_{ix}q_{jy}q_{kx}q_{lz} - q_{ix}q_{jx}q_{kz}q_{ly} + \\
&\quad + q_{ix}q_{jx}q_{kz}q_{ly} + q_{iz}q_{jx}q_{ky}q_{lx} + q_{iy}q_{jx}q_{kx}q_{lz} - q_{iz}q_{jx}q_{kx}q_{ly} - q_{iy}q_{jx}q_{kz}q_{lx} - q_{ix}q_{jx}q_{ky}q_{lz} + \\
&\quad + q_{iy}q_{jz}q_{kx}q_{lx} + q_{ix}q_{jy}q_{kx}q_{lz} + q_{iz}q_{jx}q_{kx}q_{ly} - q_{iy}q_{jx}q_{kx}q_{lz} - q_{ix}q_{jz}q_{kx}q_{ly} - q_{iz}q_{jy}q_{kx}q_{lx} + \\
&\quad + q_{iz}q_{jy}q_{kx}q_{lx} + q_{iy}q_{jx}q_{kz}q_{lx} + q_{ix}q_{jz}q_{ky}q_{lx} - q_{ix}q_{jy}q_{kz}q_{lx} - q_{iz}q_{jx}q_{ky}q_{lx} - q_{iy}q_{jz}q_{kx}q_{lx} \\
&= 0 \\
\boldsymbol{\beta}^j \cdot \mathscr{y} &= q_{iy}q_{jx}q_{ky}q_{lz} + q_{iy}q_{jz}q_{kx}q_{ly} + q_{iy}q_{jy}q_{kz}q_{lx} - q_{iy}q_{jz}q_{ky}q_{lx} - q_{iy}q_{jy}q_{kx}q_{lz} - q_{iy}q_{jx}q_{kz}q_{ly} + \\
&\quad + q_{ix}q_{jy}q_{kz}q_{ly} + q_{iz}q_{jy}q_{ky}q_{lx} + q_{iy}q_{jy}q_{kx}q_{lz} - q_{iz}q_{jy}q_{kx}q_{ly} - q_{iy}q_{jy}q_{kz}q_{lx} - q_{ix}q_{jy}q_{ky}q_{lz} + \\
&\quad + q_{iy}q_{jz}q_{ky}q_{lx} + q_{ix}q_{jy}q_{ky}q_{lz} + q_{iz}q_{jx}q_{ky}q_{ly} - q_{iy}q_{jx}q_{ky}q_{lz} - q_{ix}q_{jz}q_{ky}q_{ly} - q_{iz}q_{jy}q_{ky}q_{lx} + \\
&\quad + q_{iz}q_{jy}q_{kx}q_{ly} + q_{iy}q_{jx}q_{kz}q_{ly} + q_{ix}q_{jz}q_{ky}q_{ly} - q_{ix}q_{jy}q_{kz}q_{ly} - q_{iz}q_{jx}q_{ky}q_{ly} - q_{iy}q_{jz}q_{kx}q_{ly} \\
&= 0 \\
\boldsymbol{\beta}^j \cdot \mathscr{z} &= q_{iz}q_{jx}q_{ky}q_{lz} + q_{iz}q_{jz}q_{kx}q_{ly} + q_{iz}q_{jy}q_{kz}q_{lx} - q_{iz}q_{jz}q_{ky}q_{lx} - q_{iz}q_{jy}q_{kx}q_{lz} - q_{iz}q_{jx}q_{kz}q_{ly} + \\
&\quad + q_{ix}q_{jz}q_{kz}q_{ly} + q_{iz}q_{jz}q_{ky}q_{lx} + q_{iy}q_{jz}q_{kx}q_{lz} - q_{iz}q_{jz}q_{kx}q_{ly} - q_{iy}q_{jz}q_{kz}q_{lx} - q_{ix}q_{jz}q_{ky}q_{lz} + \\
&\quad + q_{iy}q_{jz}q_{kz}q_{lx} + q_{ix}q_{jy}q_{kz}q_{lz} + q_{iz}q_{jx}q_{kz}q_{ly} - q_{iy}q_{jx}q_{kz}q_{lz} - q_{ix}q_{jz}q_{kz}q_{ly} - q_{iz}q_{jy}q_{kz}q_{lx} + \\
&\quad + q_{iz}q_{jy}q_{kx}q_{lz} + q_{iy}q_{jx}q_{kz}q_{lz} + q_{ix}q_{jz}q_{ky}q_{lz} - q_{ix}q_{jy}q_{kz}q_{lz} - q_{iz}q_{jx}q_{ky}q_{lz} - q_{iy}q_{jz}q_{kx}q_{lz} \\
&= 0.
\end{aligned}$$

$\square$

Once the DA methodology has obtained the $B_{\mathbf{p}}$ set, it needs to compute the final set of coordinates $A = \{\alpha_1, \ldots, \alpha_n\}$ that reproduces the point $\mathbf{p}$. First, the DA normalises the vectors $\boldsymbol{\beta}^j \in B_{\mathbf{p}}$, so

$$\boldsymbol{\beta}^{j'} = \frac{\boldsymbol{\beta}^j}{\|\boldsymbol{\beta}^j\|}.$$

Then, it weighs them by means of the cardinality of the set $B_{\mathbf{p}}$. That is

$$\omega_i = \frac{1}{|B_{\mathbf{p}}|} \sum_{\boldsymbol{\beta}^j \in B_{\mathbf{p}}} \beta_i^{j'}.$$

Now, we get the set of coordinates $\Omega = \{\omega_1, \ldots, \omega_n\}$ such that

$$\sum_{i=1}^{n} \omega_i \vec{\mathbf{q}}_i = \mathbf{0}.$$

Finally, the DA normalises the Ω set

$$\mathbf{0} = \sum_{i=1}^{n} \omega_i \vec{\mathbf{q}}_i = \sum_{i=1}^{n} \omega_i \frac{\mathbf{v}_i - \mathbf{p}}{\|\mathbf{v}_i - \mathbf{p}\|},$$

so

$$\alpha_i = \frac{\frac{\omega_i}{\|\mathbf{v}_i - \mathbf{p}\|}}{\sum_{j=1}^{n} \frac{\omega_j}{\|\mathbf{v}_j - \mathbf{p}\|}}.$$

Although this weighting technique results in a positive set of coordinates, the weight $\frac{1}{|B_{\mathbf{p}}|} \in \mathbb{Q}$ is discrete and does not provide the required continuity for the functions of the coordinates. Figure 3.34

shows a plot of the coordinates for the classical model of the line surrounded by nine control points. The discontinuities shown in the functions of the coordinates are caused when neighbouring points $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ of the model have a different number of configurations in their corresponding sets $B_{\mathbf{p}_1}$, $B_{\mathbf{p}_2}$ and $B_{\mathbf{p}_3}$. This situation causes that the influence of the control points changes drastically between them due to the discrete nature of the weight $\frac{1}{|B_{\mathbf{p}}|}$.

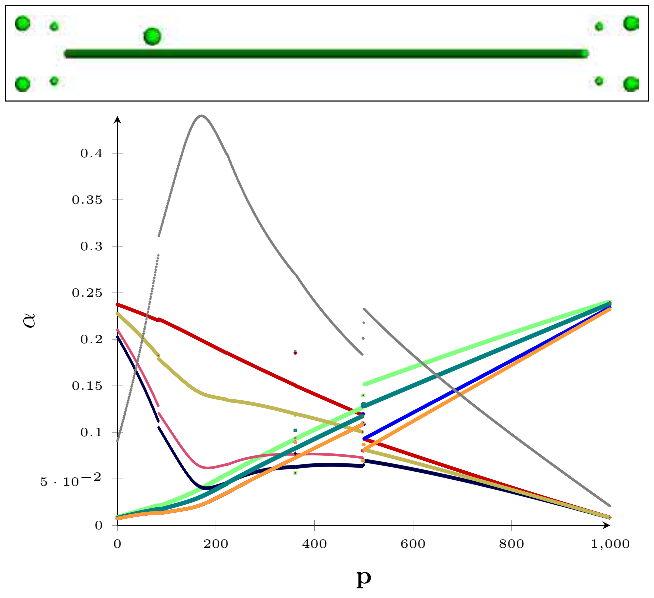


Figure 3.34: Discontinuities arise on the DA coordinate functions.

Figure 3.35 presents a typical situation where a discontinuity occurs. While in $\mathbf{p}_1$ the weight is $\frac{1}{|B_{\mathbf{p}_1}|} = \frac{1}{2}$, in $\mathbf{p}_2$ is $\frac{1}{|B_{\mathbf{p}_2}|} = \frac{1}{4}$ and in $\mathbf{p}_3$ is $\frac{1}{|B_{\mathbf{p}_3}|} = \frac{1}{3}$. This fact takes place because the DA does not perform a partition of the convex hull of the control points. Instead, the tetrahedra can overlap with each other. This means that asymmetries can appear between both sides of some of the faces shared among several tetrahedra, where we describe an asymmetry as the situation where a face does not have the same number of tetrahedra on its left side as on its right side.

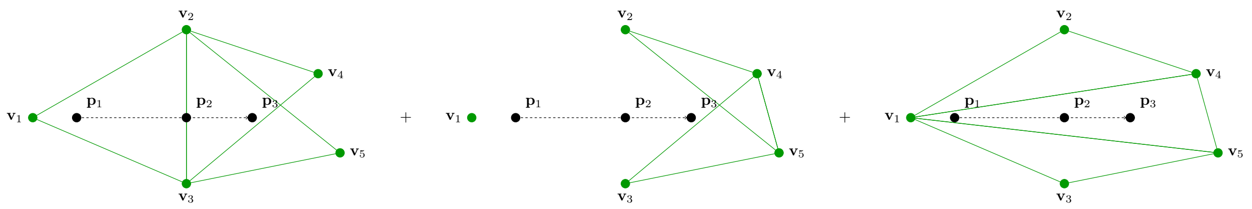


Figure 3.35: Example of a transition that causes a discontinuity in the functions of the coordinates. Notice that all the possible triangles are not disjointed. This means that the edge $\overline{\mathbf{v}_2\mathbf{v}_3}$ has one left triangle and two right triangles. Such asymmetry causes discontinuous coordinates.

These discontinuities make the method unsuitable to be applied in deformation processes. So, we finally rejected the DA to move forward, looking for new ways to compute coordinate systems.

3.3.4 Gradient Method Approach

During the last experiments we have presented, especially the Lagrange Minimisation Approach (LMA) and the Wire Approach (WA), we experienced that reaching the $V^{\perp} \cap \mathbb{R}^n_+$ subspace was not an easy task. However, an alternative method was to take an initial point in $V^{\perp}$ and make it evolve inside the same subspace $V^{\perp}$ until it reached the region $\mathbb{R}^n_+$. In this way, the modified point could be used as the coordinates of a particular model point $\mathbf{p}$ with respect to the control points.

The Gradient Method Approach (GMA) maximises a positive attracting function. It starts projecting a random point $p \in \mathbb{R}^n_+$ to the linear subspace $V^{\perp}$. We call the projected point as $\bar{p}$ and it is computed by means of the projection matrix $P_{V^{\perp}}$ (refer to Annex A). Then, the GMA iteratively translates $\bar{p}$ over $V^{\perp}$ using the gradient method, as its own name suggests, to bring it to $V^{\perp} \cap \mathbb{R}^n_+$.

The gradient method consists of, given a point q, translating it through the direction of its maximum change, that is, the gradient $\vec{\mathbf{g}}$. Hence, the GMA moves the point $\bar{p}$ through its gradient over $V^{\perp}$, which we call $\vec{\mathbf{g}}_{V^{\perp}}$.

In order to compute $\vec{\mathbf{g}}_{V^{\perp}}$, we use the function $\mathrm{minC}(\bar{p})$ ("minimum coordinate"), which returns the value of the lowest coordinate in $\bar{p}$ and, also, the index, or indices, of such coordinate. Then, and assuming that $\mathrm{minC}(\bar{p})$ function returns the index i as the one with the minimum coordinate value in $\bar{p}$, the GMA defines an appropiate gradient $\vec{\mathbf{g}}$ as the vector

$$\begin{cases} g_i &= 1.0 \\ g_j &= 0.0 \quad \forall j \neq i. \end{cases}$$

However, the gradient $\vec{\mathbf{g}}$ lies in $\mathbb{R}^n$, so it needs to be projected to $V^{\perp}$ before it can be applied: $\vec{\mathbf{g}}_{V^{\perp}} = \vec{\mathbf{g}} \cdot P_{V^{\perp}}$. Now, GMA can translate $\bar{p}$ by means of $\vec{\mathbf{g}}_{V^{\perp}}$

$$\bar{p}^1 = \bar{p} + \delta_1 \cdot \vec{\mathbf{g}}_{V^{\perp}},$$

where δ_1 is the distance that $\bar{p}$ can be moved until the lowest coordinate ties with the next coordinate, or coordinates. When this occurs, GMA checks that $\mathrm{minC}(\bar{p}^1) > \mathrm{minC}(\bar{p})$, that is, the lowest coordinate after the first iteration of the gradient method is higher than the lowest coordinate of the original point $\bar{p}$. If that happens, the translation can be applied. If not, the GMA seeks another coordinate to tie with.

Once the first tie is reached, we get the point $\bar{p}^1$, which has two or more coordinates with the same lowest value. Let us imagine that $\mathrm{minC}(\bar{p}^1)$ returns the indices (i, j, k) as the lowest coordinates in $\bar{p}^1$. Now, GMA moves to the next iteration of the method and computes the gradient $\vec{\mathbf{g}} \in \mathbb{R}^n$ as

$$\begin{cases} g_i &= 1.0 \\ g_j &= 1.0 \\ g_k &= 1.0 \\ g_l &= 0.0 \quad \forall l \neq i, j, k. \end{cases}$$

This time, $\vec{\mathbf{g}}$ must be projected over $V^{\perp}_{ijk}$, where $V^{\perp}_{ijk}$ is the subspace of $V^{\perp}$ where the coordinates corresponding to the indices i, j and k are equal. Now, the projection matrix has changed and our method needs to expand the original matrix $P_{V^{\perp}}$ in order to define $P_{V^{\perp}_{ijk}}$. It first adds the new conditions to the matrix V (see Annex A)

$$\begin{aligned} v_{ij} &= (\underbrace{1}_{i},\dots,0,\dots,\underbrace{-1}_{j},\dots,0) && \Rightarrow i=j \\ v_{ij} &= (0,\dots,\underbrace{1}_{j},\dots,0,\dots,\underbrace{-1}_{k}) && \Rightarrow j=k, \end{aligned}$$

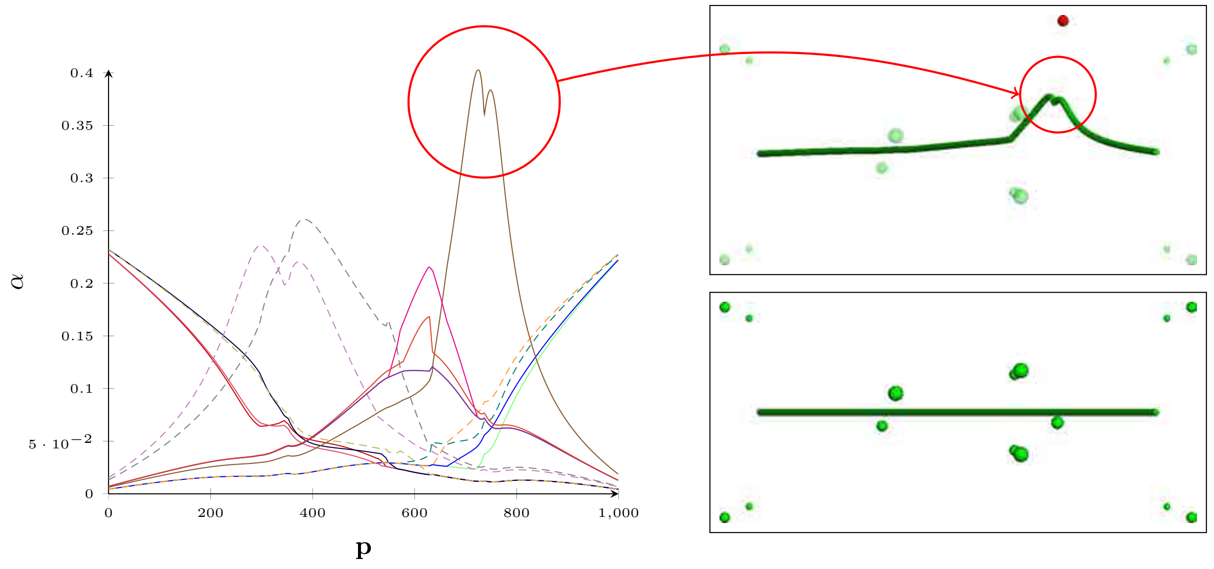
so $V_{ijk} = \begin{pmatrix} x \\ y \\ z \\ v_{ij} \\ v_{jk} \end{pmatrix}$. Then, the new system of equations is

$$(p - \bar{p}) = \rho_x x + \rho_y y + \rho_z z + \rho_{v_{ij}} v_{ij} + \rho_{v_{jk}} v_{jk}. \tag{3.31}$$

Notice that $v_{ij} \cdot \bar{p} = v_{jk} \cdot \bar{p} = 0$, so we can also multiply Equations (3.31) by the vectors x, y, z, v_{ij} and v_{jk} to find the matrix M_{ijk}, similarly as in Annex A, Section A.2. Once M_{ijk} is computed, the new projection matrix $P_{V_{ijk}^{\perp}}$ is defined as $P_{V_{ijk}^{\perp}} = I - V_{ijk}^{\top} \cdot M_{ijk}^{-1} \cdot V_{ijk}$. Now, GMA can translate the point, so

$$\bar{p}^{i} = \bar{p}^{i-1} + \delta_i \cdot \vec{\mathbf{g}}^{V_I^{\perp}},$$

where the label i indicates the current iteration and $V_I^{\perp}$ is the current subspace in $V^{\perp}$. The gradient method ends when the function $\min\mathrm{C}(\bar{p}^{i})$ does not increase anymore.



(a) (b)

Figure 3.36: Figure (a) shows the GMA coordinate functions corresponding to the initial configuration of a line surrounded by fifteen control points. The bottom image in (b) presents the initial configuration while the top image shows a simple deformation produced by the translation of the handle in red. The red circles highlight an area where the problems caused by the low order of continuity of the coordinates arise.

The GMA coordinates are well-defined, positive and functional. However, they present an important drawback: every time that two or more coordinates tie, the smoothness of the functions of the coordinates decreases, so the GMA coordinates are C^0-continuous only, as we can see in Figure 3.36. This low order of continuity results in rough deformations that produce visual artifacts and unpleasant shapes on the deformed models, as shown in Figure 3.37. Consequently, we also discarded the GMA method because it was not appropriate enough to be applied to any deformation framework.

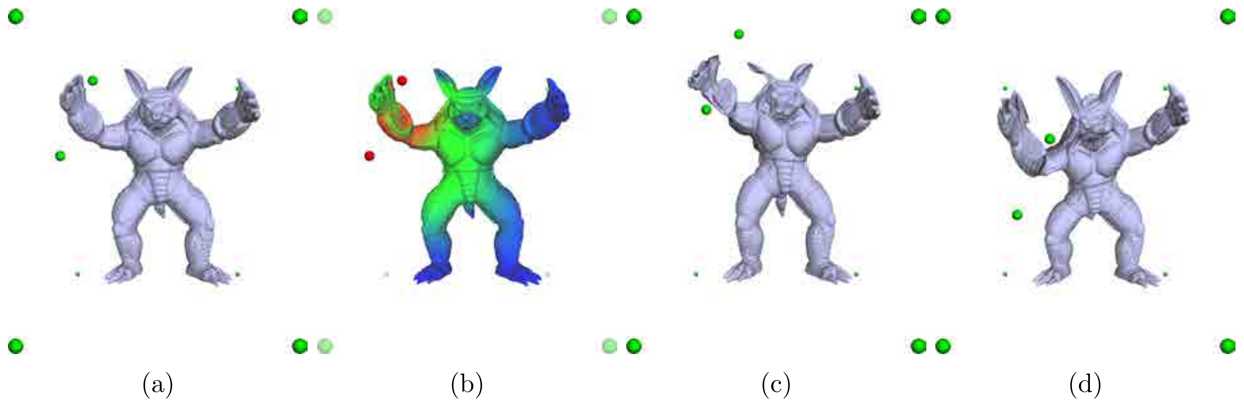


Figure 3.37: Example of a deformation and the corresponding heat map using the GMA coordinates. (a) shows the initial configuration of the *Armadillo* model surrounded by ten control points. (b) presents the heat map that corresponds to the handles in red. Finally, (c) and (d) illustrate two deformations that arise the problems caused by the low order of continuity of the GMA coordinates.

3.3.5 Conclusions

In this section we have studied several methods that try to build a well-defined system of coordinates in the n-Dimensional Space. They do not need any kind of connectivity or structure defined over the control points, but only their location in the 3D space around the model to be deformed.

The presented approaches represent the control points in n-Dimensional Space. Evenmore, we have bounded the region of the nD space in which the coordinates that fulfil all our requirements lie. This region is $V^{\perp} \cap \mathbb{R}^n_+$.

We have described four methods that do not always achieve good results. On the one hand, the Lagrange Minimisation Approach (LMA) and the Wire Approach (WA) show us that reaching the $V^{\perp} \cap \mathbb{R}^n_+$ is not a simple task. On the other hand, the Determinants Approach (DA) and the Gradient Method Approach (GMA) present either discontinuities or low order of continuity that make them useless for deformation processes.

As far as the obtained results do not achieve the required objectives, we need to look for new methods that solve the drawbacks of the analysed approaches.

3.4 Summary

Our main goal is to go beyond the limitations of most of the existing cage-based deformation schemes. We want to increase their flexibility and to simplify their handles to make the deformation processes more intuitive and easy to drive. In this way, this chapter has been presented as a compilation of experiments and attempts to compute a well-defined system of coordinates. Most of the presented approaches are not suitable to be applied in real deformation frameworks due to their inefficiency and negativity and continuity problems. However, they show how the problem of defining a set of coordinates can be dealt with different and interesting points of view. In addition, this chapter is also a good reference to understand how our ideas and hypothesis have evolved during the research process.

Firstly, we present the Mean Value Coordinates Based Methods, which include the Delaunay Triangulation Approach (DTA), the Voronoi Regions Triangulation Approach (VRTA) and the Incremental Approach (IA). In the DTA and the VRTA we work with Delaunay Triangulations and Voronoi Diagrams to dynamically and automatically compute the connectivity between the control points. Nevertheless, these methods do not provide the continuity requirement over the coordinates due to the strong dependency of the MVC on the supporting structure defined over the control points, that is, their connectivity (see Figure 3.5). Although we can avoid this drawback by combining the use of the Laguerre Voronoi Diagram over the sphere and the computation of the Natural Neighbour Coordinates (NNC) on the plane, the resulting technique is not efficient enough. On the other hand, the IA makes distinction between exterior and interior control points. Firstly, it computes a Delaunay Triangulation of the convex hull of the exterior control points. Then, it adds the rest of the interior control points one by one and updates the coordinates. Although the resulting coordinates are continuous and smooth, the IA method presents negative values as the number of control points increases (see Figure 3.15).

After these first experiments, we realise that the discontinuities on the coordinate functions are caused by the dynamic changes on the connectivity of the control points. So, we decide to establish an automatic and static connectivity. This connectivity of the control points is the supporting structure over which we compute the coordinates for all the points of the 3D model. We base this static connectivity on the computation of tetrahedra. However, all the obtained results, both numerical and analytical, present a low order of continuity. This means that the deformations driven by these kind of coordinates show artifacts and sharp features on the deformed models (see Figures 3.21 and 3.23). Consequently, we also discard these techniques.

Finally, we want to study the problem directly from the n-Dimensional Space, where n is the number of control points. We avoid any connectivity between the control points to increase the smoothness of our coordinates. In this context, we study four different techniques, the Lagrange Minimisation Approach (LMA), the Wire Approach (WA), the Determinants Approach (DA) and the Gradient Method Approach (GMA). Although none of them give satisfactory results (see Figures 3.27, 3.34 and 3.36), they open a new perspective to approach the problem of building a well-defined system of coordinates. It is this new perspective what we will exploit in Chapter 4 to develop the C-Celestial Coordinates (CCC), the T-Celestial Coordinates (TCC) and the S-Celestial Coordinates (SCC). The n-Dimensional based approaches studied in the present chapter lack the continuity and the smoothness required for the deformation processes. In contrast, the CCC and, especially, the TCC and the SCC are well-defined systems of coordinates, continuous and smooth that can be perfectly applied in any deformation application. As we will see in Chapter 4, the deformations driven by the TCC and the SCC produce good and visually pleasant results over the deformed models and are intuitive and easy to use.

Chapter 4

Point-Based Celestial Coordinates

"Eureka!"
Archimedes

Chapter 4 Contents

In the previous chapter, we presented different approaches to the cage-free deformation problem which can be classified into three different groups: the *Mean Value Coordinates Based Methods*, the *Layered Based Methods* and the n*-Dimensional Space Based Methods*. However, none of them meets our expectations, i.e. they do not fulfil neither the basic properties listed in the introduction of Chapter 3 nor the good features enunciated in Section 1.1. The described Mean Value Based Methods suffer from discontinuous and negative coordinate functions or from complex and inefficient computations, despite the correctness of the final set of coordinates. The Layered Based Methods result in rough coordinates that do not have the required level of smoothness and locality. Finally, the studied n-Dimensional Space Based Methods also present discontinuous, negative and sharp coordinates. Moreover, they are not always successful in finding a set of coordinates for every point $\mathbf{p}$ of the model.

Since the results presented so far were not as promising as desired, we still wanted to find a well-defined system of coordinates which could be used in real deformation frameworks. We, therefore, continued our research until we could, finally, define some successful techniques. Accordingly, this chapter is devoted to detail, analyse and discuss the new proposed coordinate systems.

Firstly, we define a new family of coordinates in Section 4.1, the *Celestial Coordinates Family*, which comprises some of the existing schemes and the methods presented in the previous chapter.

Then, we develop three new systems of coordinates: the *C-Celestial Coordinates* in Section 4.2, the *T-Celestial Coordinates* in Section 4.3 and the *S-Celestial Coordinates* in Section 4.4. They are new cage-free approaches that belong to the n-Dimensional Space Based Methods as well as to the Celestial Coordinates family. Section 4.5 studies a method to increase the locality of the coordinates. This technique is an attempt to localise the deformation by grouping subsets of control points. Hence, we want to combine the new systems of coordinates together with the localising methodology to obtain a well-defined cage-free deformation approach that satisfies all the basic properties[1]: reproduction of the identity, reproduction of the unity, positivity, continuity, smoothness and locality. Moreover, it should be robust, flexible, easy to use and have interactive response. Section 4.6 discusses the properties of the three new systems of coordinates and compares the proposed deformation techniques to some existing methodologies. Finally, Section 4.7 presents the conclusions of the chapter.

Part of the research in this chapter has been compiled into a paper which is now under review.

4.1 Celestial Coordinates Family

In this section, we introduce the Celestial Coordinates, a family that groups some Generalised Barycentric Coordinate (GBC) schemes. The systems that belong to this family are characterised by two main features. The first feature is that they are positive GBC, that is, the coordinates are positive inside the deformation domain. We define the deformation domain as the volume inside which the deformation takes place. In the classical cage-based paradigm, this domain is bounded by the faces of the cage. In contrast, in a cage-free environment, the domain is defined by the convex hull of the control points. The second feature refers to the projection of the control points over a unit sphere during their computation process. We call this the *celestial sphere*, hence the name of the coordinate family.

Given a set of control points $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ and a point $\mathbf{p}$ inside the deformation domain, any system of Celestial Coordinates computes the coordinates $A = \{\alpha_1, \ldots, \alpha_n\}$ of $\mathbf{p}$ with respect to the control points $\mathbf{v}_i$ by means of the resolution of the well-known linear combination

$$\mathbf{p} = \sum_{i=1}^{n} \alpha_i(\mathbf{p})\mathbf{v}_i.$$

Based on the same principles as the Mean Value Coordinates [FKR05] and adding the requirement of positiveness, these schemes first project the handles $\mathbf{v}_i$ onto $\mathcal{S}_{\mathbf{p}}$, the unit sphere, or *celestial sphere*, centered on $\mathbf{p}$, and solve the simplified linear combination

$$\sum_{i=1}^{n} \beta_i \vec{\mathbf{q}}_i = \vec{\mathbf{0}} \qquad \forall i \;\; \beta_i \geq 0, \tag{4.1}$$

where $\vec{\mathbf{q}}_i = \frac{(\mathbf{v}_i - \mathbf{p})}{d_i}$, for $d_i = \|(\mathbf{v}_i - \mathbf{p})\|$. Therefore, the final coordinates α_i are obtained as

$$\alpha_i(\mathbf{p}) = \frac{\omega_i(\mathbf{p})}{\sum_{j=1}^{n} \omega_j(\mathbf{p})}, \tag{4.2}$$

where $\omega_i(\mathbf{p}) = \frac{\beta_i}{d_i}$.

[1]Refer to the introduction of Chapter 3 for a detailed description of each property.

The main difference between the Celestial Coordinate systems and the Mean Value Coordinates (MVC) is that the schemes included in the Celestial Coordinates family ensure the positivity of the β_i coordinates inside all the convex hull of the control points. Hence, $\alpha_i \in V^\perp \cap \mathbb{R}^n_+$, as they are defined in Equations (3.18) and (3.19). Instead, the MVC are only positive if $\mathbf{p}$ is located inside the kernel of the user-defined cage. Thus, they behave as Celestial Coordinates if, and only if, the cage is convex. A similar situation takes place with the Spherical Barycentric Coordinates [LBS06]. On the contrary, the Positive Mean Value Coordinates [LKCOL07] are a good example of a member of the Celestial Coordinates family since they are always positive, even inside star-shaped cages.

The reader must have observed that not all the schemes presented in Chapter 3 succeed in their goal to become a well-defined positive GBC system, hence they do not belong to the family of Celestial Coordinates. Despite this fact, we are particularly interested in those techniques defined in Section 3.3 as the n-Dimensional Space Based Methods, where n is the number of user-defined deformation handles. Their main feature is that they depend neither on a user-defined nor an automatically computed connectivity between the control points, but they only take into account the position of these handles. Therefore, we classify all the positive n-Dimensional Space Based GBC systems into a subfamily called the *Point-Based Celestial Coordinates*. That is, the Point-Based Celestial Coordinates collect all the schemes in the Celestial Coordinates family that do not need any kind of connectivity between the control points to compute a set of coordinates.

The n-dimensional methodologies presented in the previous chapter, which are the *Lagrange Minimisation Approach*, the *Wire Approach*, the *Determinants Approach* and the *Gradient Method Approach*, experience the important disadvantages of negativity and discontinuity. We needed to address the problem from a new angle. We wanted to find an easier and more appropriate way to compute a vector $\boldsymbol{\beta} \in V^\perp \cap \mathbb{R}^n_+$ and, therefore, obtain a correct Point-Based Celestial Coordinate system to relate the points $\mathbf{p}$ of the model with respect to the deformation handles. Accordingly, the following sections present three new schemes that belong to this subgroup of the Celestial Coordinates family. They are the C-Celestial Coordinates, the T-Celestial Coordinates and the S-Celestial Coordinates and, unlike the methods in Section 3.3, they are well-defined.

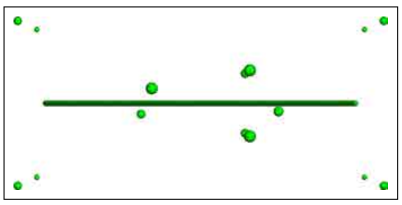
4.2 C-Celestial Coordinates

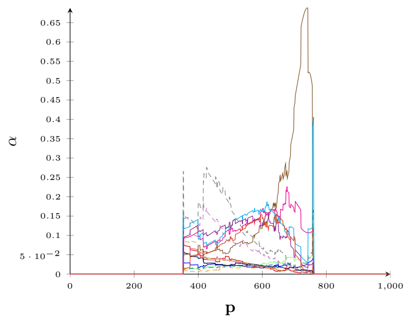
An obvious and intuitive choice to compute a valid vector $\boldsymbol{\beta}$ that satisfies Equation (4.1), while lying in the subspace $V^\perp \cap \mathbb{R}^n_+$, is to find the centroid of the intersection $V^\perp \cap \mathbb{R}^n_+ \cap \mathcal{S}$, where $\mathcal{S}$ is the n-dimensional unit hypersphere or hypercube centered at the origin $\mathbf{0} \in \mathbb{R}^n$.

Although the idea is quite simple, the analytic computation of the centroid is not an easy and evident task. Therefore, we propose to compute this centroid by means of the Monte-Carlo method. Firstly, we randomly sample the unit hypersphere or hypercube $\mathcal{S}$. Then, the sampled points are projected into $V^\perp$ by means of the projection matrix $P_{V^\perp}$ described in Annex A. Only those projected samples lying in $\mathbb{R}^n_+$, that is, with all their coordinates positive, are considered to compute the final centroid.

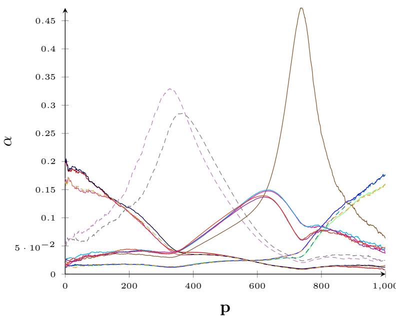
4.2.1 Discussion

The C-Celestial Coordinates are positive and well-behaved. However, the required number of Monte-Carlo samples increases overwhelmingly as the n-dimensional space also grows. That is, the larger the number n of control points is, the more samples the C-Celestial Coordinates method has to take into account to finally compute a well-defined set of coordinates for the model point $\mathbf{p}$, as shown in Figure 4.1. This disadvantage results in unaffordable computational costs, making this technique useless for practical applications (see Figure 4.18 in Section 4.6.1).

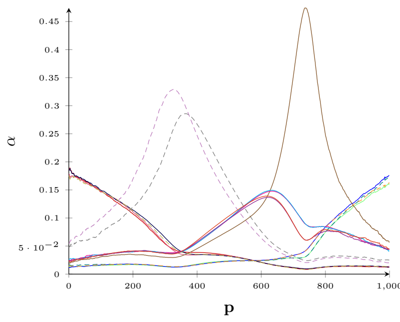




(a) 10^5 samples



(b) $25 \cdot 10^7$ samples



(c) 10^9 samples

Figure 4.1: The C-Celestial Coordinate functions corresponding to the initial configuration of a line surrounded by fifteen control points, shown at the top. (a), (b) and (c) show a sequence of the coordinates which have been obtained with different number of Monte-Calro samples. Notice how the coordinate functions are softened when the number of samples increases.

4.3 T-Celestial Coordinates

As stated in the previous section, the C-Celestial Coordinates suffer from inefficient computation cost. Therefore, we researched into faster and more direct ways to compute a valid vector $\boldsymbol{\beta}$.

Given a particular point $\mathbf{p}$ of the model, the idea behind the T-Celestial Coordinates was to work with the span of the vectors $\chi, y, z \in \mathbb{R}^n$, which have been presented in Equation (3.16). Any v in the span of these three vectors depends on three unknowns $\boldsymbol{\psi} = (\psi_\chi, \psi_y, \psi_z)$, such that

$$v = \psi_\chi \chi + \psi_y y + \psi_z z.$$

Hence, the vector $\boldsymbol{\beta}$ and the final set of coordinates for $\mathbf{p}$ also depend on the triplet $(\psi_\chi, \psi_y, \psi_z)$.

The proposed method consists in, first of all, applying a transformation function $T(\psi_\chi, \psi_y, \psi_z)$ to

v, which lends the name to the present coordinates. Such transformation sends v to $\mathbb{R}^n_+$

$$\begin{array}{rlcl} T: & \mathbb{R}^n & \to & \mathbb{R}^n_+ \\ & v = \psi_\chi \chi + \psi_y y + \psi_z z & \to & T(\psi_\chi, \psi_y, \psi_z). \end{array} \tag{4.3}$$

Then, we get $v' = T(\psi_\chi, \psi_y, \psi_z)$ lying in $\mathbb{R}^n_+$, so we must still ensure that it also lies in $V^\perp$. In other words, we need to solve the system of equations

$$\begin{aligned} T(\psi_\chi, \psi_y, \psi_z) \cdot \chi &= 0 \\ T(\psi_\chi, \psi_y, \psi_z) \cdot y &= 0 \\ T(\psi_\chi, \psi_y, \psi_z) \cdot z &= 0, \end{aligned} \tag{4.4}$$

for the unknowns $(\psi_\chi, \psi_y, \psi_z)$, which is equivalent to the previous System (3.17). Once the System (4.4) is solved, the vector $\boldsymbol{\beta}$, over the unit sphere $\mathcal{S}_\mathbf{p}$ centered on $\mathbf{p}$, is

$$\boldsymbol{\beta} = T(\psi_\chi, \psi_y, \psi_z),$$

for the particular triplet $(\psi_\chi, \psi_y, \psi_z)$ that satisfies it. The final set of coordinates $\alpha_i(\mathbf{p})$ are computed as done in [FKR05] and shown in Equation (4.2).

In the described procedure there are two main points that need our attention. The first one is the definition of the transformation function $T(\psi_\chi, \psi_y, \psi_z)$. The second one, the methodology to solve the System (4.4). The next two sections are, therefore, devoted to analysing both points and detail the solution that we propose.

4.3.1 The Transformation Function $T(\psi_\chi, \psi_y, \psi_z)$

As exposed previously and illustrated in Expression (4.3), we wanted to define a transformation function $T(\psi_\chi, \psi_y, \psi_z)$ such that it sends the vector

$$v = \psi_\chi \chi + \psi_y y + \psi_z z$$

to $\mathbb{R}^n_+ \subset \mathbb{R}^n$. Hence, for any vectors χ, y and z and for any triplet $\boldsymbol{\psi} = (\psi_\chi, \psi_y, \psi_z)$, the transformation $T(\psi_\chi, \psi_y, \psi_z)$ must ensure that her image belongs to $\mathbb{R}^n_+$.

There are many functions whose images live in $\mathbb{R}^n_+$. For example,

$$\begin{aligned} f_1(x) &= \mathrm{e}^x \\ f_2(x) &= x^\nu \qquad \text{where } \nu \in \mathbb{N} \ \wedge \ \nu = 2k. \end{aligned}$$

In particular, we defined our transformation using the exponential function f_1. Thus, we propose

$$T(\psi_\chi, \psi_y, \psi_z) = \left(\mathrm{e}^{\psi_\chi \chi_1 + \psi_y y_1 + \psi_z z_1}, \ldots, \mathrm{e}^{\psi_\chi \chi_n + \psi_y y_n + \psi_z z_n}\right).$$

This expression can be simplified into

$$T(\psi_\chi, \psi_y, \psi_z) = \left(\mathrm{e}^{\boldsymbol{\psi} \cdot \vec{\mathbf{q}}_1}, \ldots, \mathrm{e}^{\boldsymbol{\psi} \cdot \vec{\mathbf{q}}_n}\right), \tag{4.5}$$

where $\boldsymbol{\psi} = (\psi_\chi, \psi_y, \psi_z)$ and $\vec{\mathbf{q}}_i = (\chi_i, y_i, z_i)$ from Equality (3.30). Additionally, a particular user may also want to establish specific weights for each of the deformation handles, so their influence regions can be adapted. In order to add this new property, we modified the original transformation function in Equation (4.5) to take into account an initial fixed user-defined point $s \in \mathbb{R}^n_+ \cup \{\mathbf{0}\}$. Hence, the final transformation function is the following

$$T(\psi_\chi, \psi_y, \psi_z) = \sum_{i=1}^{n} s_i + \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}. \tag{4.6}$$

4.3.2 Solving for $T(\psi_\chi, \psi_y, \psi_z)$

In the section above, we have defined the transformation $T(\psi_\chi, \psi_y, \psi_z)$ to send $v = \psi_\chi \chi + \psi_y y + \psi_z z$ to $\mathbb{R}^n_+$. Now, it is time to solve the System (4.4) when $T(\psi_\chi, \psi_y, \psi_z)$ is the one from Equation (4.6). However, we must first prove that this system always has a solution. The replacement of Equation (4.6) into System (4.4) leads to the new arranged system

$$\sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}\vec{\mathbf{q}}_i = \mathbf{r}, \tag{4.7}$$

where $\mathbf{r} = -\sum_{i=1}^{n} s_i \vec{\mathbf{q}}_i$. Hence, we have to demonstrate that $\forall \mathbf{r} \in \mathbb{R}^3, \exists \boldsymbol{\psi} = (\psi_\chi, \psi_y, \psi_z)$ such that the System (4.7) is solved.

Proof. Let us define the function $g(\boldsymbol{\psi}) = \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}\vec{\mathbf{q}}_i$ from $\mathbb{R}^3$ to $\mathbb{R}^3$. We need to demonstrate that it is surjective.

We first compute the partial derivatives of $g(\boldsymbol{\psi})$ as follows

$$\frac{\partial g(\boldsymbol{\psi})}{\partial \psi_\chi} = \sum_{i=1}^{n} q_{i_x}\mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}\vec{\mathbf{q}}_i, \qquad \frac{\partial g(\boldsymbol{\psi})}{\partial \psi_y} = \sum_{i=1}^{n} q_{i_y}\mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}\vec{\mathbf{q}}_i, \qquad \frac{\partial g(\boldsymbol{\psi})}{\partial \psi_z} = \sum_{i=1}^{n} q_{i_z}\mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}\vec{\mathbf{q}}_i.$$

Then, the corresponding jacobian matrix $J(\boldsymbol{\psi})$ is the one shown below

$$J(\boldsymbol{\psi}) = \begin{pmatrix} \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} q_{i_x} q_{i_x} & \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} q_{i_y} q_{i_x} & \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} q_{i_z} q_{i_x} \\ \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} q_{i_x} q_{i_y} & \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} q_{i_y} q_{i_y} & \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} q_{i_z} q_{i_y} \\ \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} q_{i_x} q_{i_z} & \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} q_{i_y} q_{i_z} & \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} q_{i_z} q_{i_z} \end{pmatrix}, \tag{4.8}$$

which is a real symmetric matrix. Hence, it has a full set of orthonormal eigenvectors with real and positive eigenvalues. Notice that if $\vec{\mathbf{v}}$ is a unit eigenvector with its corresponding eigenvalue λ, then

$$\begin{aligned} \lambda &= \vec{\mathbf{v}} J(\boldsymbol{\psi}) \vec{\mathbf{v}}^\top \\ &= \left(\sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}(\vec{\mathbf{v}}\cdot\vec{\mathbf{q}}_i^\top) q_{i_x} \quad \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}(\vec{\mathbf{v}}\cdot\vec{\mathbf{q}}_i^\top) q_{i_y} \quad \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}(\vec{\mathbf{v}}\cdot\vec{\mathbf{q}}_i^\top) q_{i_z} \right) \vec{\mathbf{v}}^\top \\ &= \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}(\vec{\mathbf{v}}\cdot\vec{\mathbf{q}}_i^\top)(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{v}}^\top) \\ &= \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}(\vec{\mathbf{v}}\cdot\vec{\mathbf{q}}_i^\top)^2. \end{aligned} \tag{4.9}$$

Now consider a function G such that

$$\begin{array}{rlcl} G: & \mathcal{S}_{\mathbf{p}} & \rightarrow & \mathbb{R} \\ & \mathbf{g} & \rightarrow & \max_i(\mathbf{g}\cdot\vec{\mathbf{q}}_i). \end{array}$$

On the one hand, $\vec{\mathbf{q}}_i$ represent the control points projected over the unitary sphere of $\mathbf{p}$, $\mathcal{S}_{\mathbf{p}}$. They are distributed around the origin, therefore $G(\mathbf{g}) > 0$, $\forall \mathbf{g} \in \mathcal{S}_{\mathbf{p}}$. On the other hand, the sphere $\mathcal{S}_{\mathbf{p}}$ is compact, so there exists an $\epsilon > 0$ such that $G(\mathbf{g}) \geq \epsilon > 0$, $\forall \mathbf{g} \in \mathcal{S}_{\mathbf{p}}$. Hence,

$$\forall \mathbf{g} \in \mathcal{S}_{\mathbf{p}},\ \exists \vec{\mathbf{q}}_i \text{ such that } \mathbf{g}\cdot\vec{\mathbf{q}}_i \geq \epsilon.$$

If we use this expression in Equation (4.9), then

$$\lambda = \vec{\mathbf{v}} J(\boldsymbol{\psi})\vec{\mathbf{v}}^\top \geq \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}\epsilon^2 \geq \epsilon^2 \mathrm{e}^{\epsilon\|\boldsymbol{\psi}\|} > 0.$$

Thus, $\|J(\boldsymbol{\psi})\vec{\mathbf{v}}^\top\| \geq \epsilon' > 0$ for any $\boldsymbol{\psi} \in \mathbb{R}^3$ and for any eigenvector $\vec{\mathbf{v}} \in \mathcal{S}_{\mathbf{p}}$. This shows that the range of the function $g(\boldsymbol{\psi})$ is an open non-empty subset of $\mathbb{R}^3$ which has no boundary, hence, is all $\mathbb{R}^3$. Therefore, $g(\boldsymbol{\psi})$ is surjective, which means that $\forall \mathbf{r} \in \mathbb{R}^3, \exists \boldsymbol{\psi} = (\psi_x, \psi_y, \psi_z)$ such that $\sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}\vec{\mathbf{q}}_i = \mathbf{r}$, as we want to demonstrate. □

Now, we know that the System (4.7) always has a solution, so we can proceed to solve it. However, the transformation function in Equation (4.6) converts the System (4.7) into a nonlinear nonconvex one. Consequently, we can not compute an analytic solution for it, so it has to be solved by means of a numerical method.

The Newton Method [Hil56] iteratively approximates the solution of nonlinear systems of equations and we propose its use to solve the System (4.7). Each iteration k of this method consists in solving the following matrix system

$$\boldsymbol{\psi}^k = \boldsymbol{\psi}^{k-1} - \left(J(\boldsymbol{\psi}^{k-1})^{-1}\right)^\top \cdot f(\boldsymbol{\psi}^{k-1}),$$

where $\boldsymbol{\psi}^{k-1}$ is the solution of the previous iteration, J is the jacobian matrix of the system, shown in Expression (4.8), and $f(\boldsymbol{\psi}^{k-1}) = \sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}^{k-1}\cdot\vec{\mathbf{q}}_i}\vec{\mathbf{q}}_i - \mathbf{r}$ is the system itself.

In order to take advantage of the Newton Method, we first need to define two important points: the first guess or approximation of the solution $\boldsymbol{\psi}^0$ and the ending condition that ensures a sufficiently accurate final value. On the one hand, in the case that concerns us, this technique reaches a precise enough solution when $\sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}\vec{\mathbf{q}}_i - \mathbf{r} \leq \epsilon$, for a given small ϵ value. Ergo, the ending condition is exactly $\sum_{i=1}^{n} \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i}\vec{\mathbf{q}}_i - \mathbf{r} \leq \epsilon$, for a proposed value $\epsilon = 10^{-6}$. On the other hand, we suggest a more elaborate procedure to accelerate the Newton Method by means of an initial guess. The main idea is to look for an initial model point $\mathbf{p}_0$, also called *seed point*, in which the Newton Method converges quickly into a correct set of coordinates. Once the seed point $\mathbf{p}_0$ has its own coordinates defined, the final solution for the triplet $\boldsymbol{\psi}_{\mathbf{p}_0} = (\psi_x, \psi_y, \psi_z)$ is spread among its neighbours, so they can use it as their initial guess to compute their own coordinates. Notice that, in this context, we establish that two points $\mathbf{p}_i$ and $\mathbf{p}_j$ are neighbours if the distance between them is smaller than a certain fixed value $\mathcal{d}$.

The key point is to define the first estimation for the Newton Method in $\mathbf{p}_0$. We know that the first two terms of the Taylor series for the exponential function e^x are $\mathrm{e}^x \simeq 1+x$. Therefore, we use this estimation to approximate the exponential term of the transformation function $T(\psi_\chi, \psi_y, \psi_z)$ as follows

$$T(\psi_\chi, \psi_y, \psi_z) = \sum_{i=1}^{n} s_i + \mathrm{e}^{\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i} \simeq \sum_{i=1}^{n} s_i + 1 + \boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i.$$

This simplification allows us to linearise the System (4.7) into $\sum_{i=1}^{n}(1+\boldsymbol{\psi}\cdot\vec{\mathbf{q}}_i)\vec{\mathbf{q}}_i = \mathbf{r}$, which can be written in matrix form as

$$\underbrace{\begin{pmatrix} \sum_{i=1}^{n}(q_{i_x}\cdot q_{i_x}) & \sum_{i=1}^{n}(q_{i_x}\cdot q_{i_y}) & \sum_{i=1}^{n}(q_{i_x}\cdot q_{i_z}) \\ \sum_{i=1}^{n}(q_{i_x}\cdot q_{i_y}) & \sum_{i=1}^{n}(q_{i_y}\cdot q_{i_y}) & \sum_{i=1}^{n}(q_{i_y}\cdot q_{i_z}) \\ \sum_{i=1}^{n}(q_{i_x}\cdot q_{i_z}) & \sum_{i=1}^{n}(q_{i_y}\cdot q_{i_z}) & \sum_{i=1}^{n}(q_{i_z}\cdot q_{i_z}) \end{pmatrix}}_{Q} \cdot \underbrace{\begin{pmatrix}\psi_\chi \\ \psi_y \\ \psi_z\end{pmatrix}}_{\boldsymbol{\psi}} = \underbrace{-\begin{pmatrix} \sum_{i=1}^{n} q_{i_x}(s_i+1) \\ \sum_{i=1}^{n} q_{i_y}(s_i+1) \\ \sum_{i=1}^{n} q_{i_z}(s_i+1)\end{pmatrix}}_{b.}$$

The resulting triplet $\boldsymbol{\psi} = (\psi_\chi, \psi_y, \psi_z)$ will be used as the initial guess $\boldsymbol{\psi}^0$ to start the iterative Newton Method. If the method does not converge quickly for the first selected point $\mathbf{p}_0$, we can randomly choose another seed point and start the process again.

Every time the method achieves a good set of coordinates for a given model point $\mathbf{p}_i$, those neighbours of $\mathbf{p}_i$ which do not yet have their coordinates computed are sorted from the nearest to the farthest to $\mathbf{p}_i$ and queued[2], as shown in Figure 4.2. We can observe that, the first time, $\mathbf{p}_i$ is exactly the seed point $\mathbf{p}_0$, hence, all its neighbours are sorted and queued. Then, each time that a point $\mathbf{p}_j$ is stacked out, the Newton Method is initialised with the triplet $\boldsymbol{\psi} = (\psi_\chi, \psi_y, \psi_z)$ of its closest neighbour among those that already have a good set of coordinates computed, as exemplified in Figure 4.3. If this initialisation is not good enough for the method to converge quickly, the process is restarted using the triplet of the second closest neighbour, and so on.

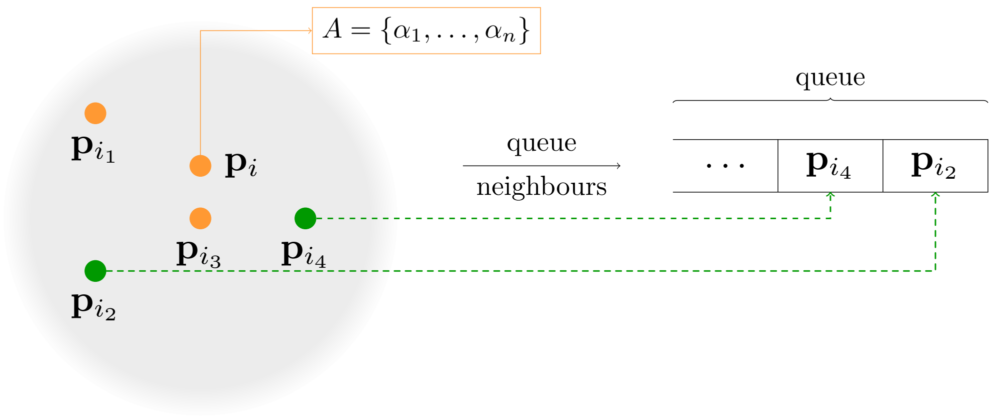


Figure 4.2: The model point $\mathbf{p}_i$ and its neighbours are displayed on the left part of the image. This point is the latest with a good set of coordinates computed. Additionally, its neighbours are divided into two groups: those with computed coordinates, in orange, and those which do not yet have coordinates, in green. The latter are queued taking into account their euclidean distance to $\mathbf{p}_i$, from the closest to the farthest, as shown on the right.

We can use two different metrics to select the closest point to $\mathbf{p}_j$. The first one is the *euclidean distance*, so the closest neighbour $\mathbf{p}$ to $\mathbf{p}_j$ is the one that minimises the distance $d_E = \|\mathbf{p} - \mathbf{p}_j\|$. The second one is the *jacobian distance*, that defines the closest neighbour $\mathbf{p}$ to $\mathbf{p}_j$ as the one that minimises the distance $d_J = \mathbf{p}_j^\top \cdot J(\boldsymbol{\psi}_\mathbf{p})^\top \cdot J(\boldsymbol{\psi}_\mathbf{p}) \cdot \mathbf{p}_j$, where $J(\boldsymbol{\psi}_\mathbf{p})$ is the jacobian matrix at the

[2]Note that if any of these neighbours is already in the queue, it is not added again.

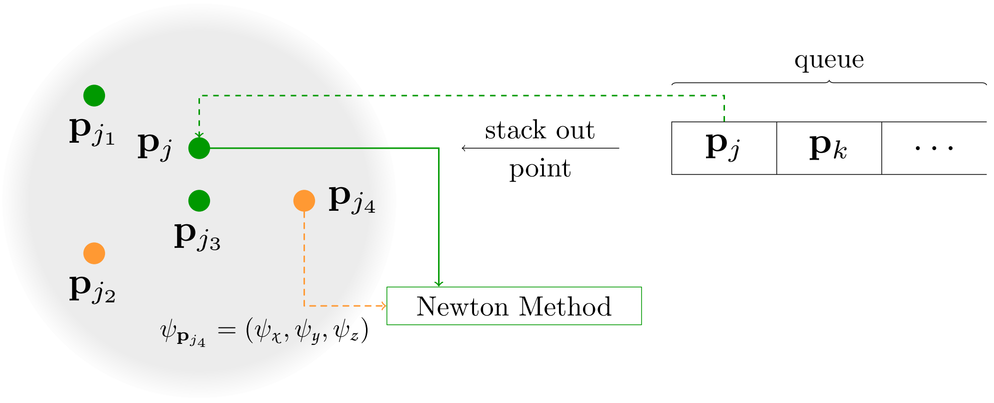


Figure 4.3: The left part of the image shows the model point $\mathbf{p}_j$ and its neighbours. This point $\mathbf{p}_j$ is located in the first position of the queue, as exposed on the right, so it is the next one for which the coordinates are to be computed. In addition, its neighbours are divided into two groups: those with computed coordinates, in orange, and those which do not yet have coordinates, in green. Therefore, the Newton Method is initialized with the triplet $\boldsymbol{\psi} = (\psi_\chi, \psi_y, \psi_z)$ of its closest neighbour between those in orange.

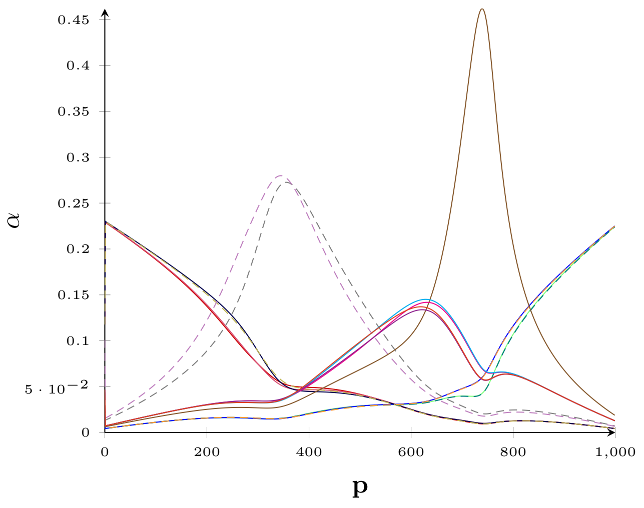
model point $\mathbf{p}_j$ by means of the triplet $\boldsymbol{\psi}_\mathbf{p} = (\psi_\chi, \psi_y, \psi_z)$, which has been obtained during the process to compute the coordinates of $\mathbf{p}$. In the next section, the analysis of the Tables 4.5 and 4.6 shows that the jacobian distance gives better results and, therefore, is the most appropriate to compute the closest point to $\mathbf{p}_j$.

4.3.3 Discussion

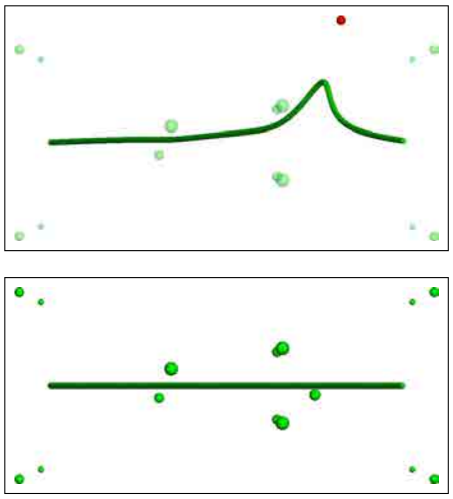
The T-Celestial Coordinates succeed in being a well-defined system of Point-Based Celestial Coordinates. They are positive, continuous and smooth, as can be seen in Figure 4.4. Furthermore, their computation is efficient despite the use of a numerical method to solve them.

This efficiency depends partly on the metric used to select the neighbour to initialise the Newton Method of a given model point $\mathbf{p}$. As presented in Section 4.3.2, we can use the euclidean distance and the jacobian distance. We, therefore, studied which of the two techniques was the most appropriate to be used during the computation of the T-Celestial Coordinates. The Newton Method together with the euclidean distance can only converge correctly and quickly with the first checked neighbour for about 50% of the points of the model, as shown in Figure 4.5. In contrast, the jacobian distance allows this methodology to converge for the first tested neighbour in 100% of the model points in our experiments. Additionally, once the most suitable neighbour has been detected, Figure 4.6 contrasts the number of iterations that the Newton Method carries out until it converges in a good set of coordinates. Although the jacobian distance needs a little more iterations, this disadvantage is largely compensated by the fact that this metric always converges for the first checked neighbour. Therefore, data presented in both figures reveal that the jacobian distance is the most convenient metric for speeding up the Newton Method.

Unfortunately, the T-Celestial Coordinates present two major drawbacks. The first one is their global effect, i.e. any deformation handle has an impact over all the points enclosed in their



(a)



(b)

Figure 4.4: The T-Celestial Coordinate functions corresponding to the initial configuration of a line surrounded by fifteen control points are shown in (a). Additionally, the bottom and the top images in (b) show this initial configuration and a simple deformation produced by the translation of the control point in red, respectively.

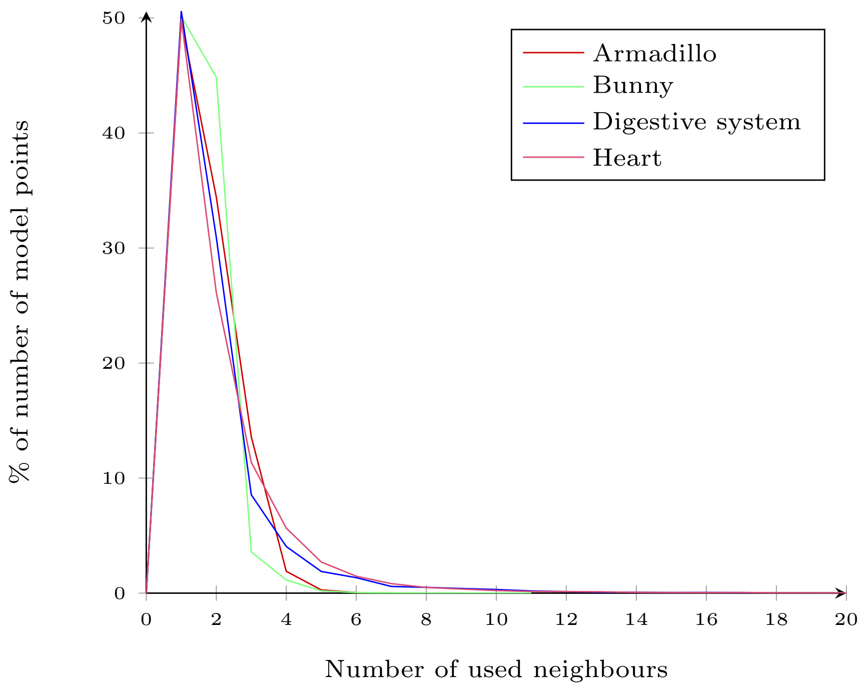


Figure 4.5: Number of neighbours that the Newton Method needs to check before it converges on a good set of T-Celestial Coordinates. In this case, the neighbours have been sorted by means of the euclidean distance. Notice that only the 50% of the points of the model can be solved with the help of the first tested *euclidean* neighbour. In contrast, the 100% of the points of the model can be solved with the first checked *jacobian* neighbour.

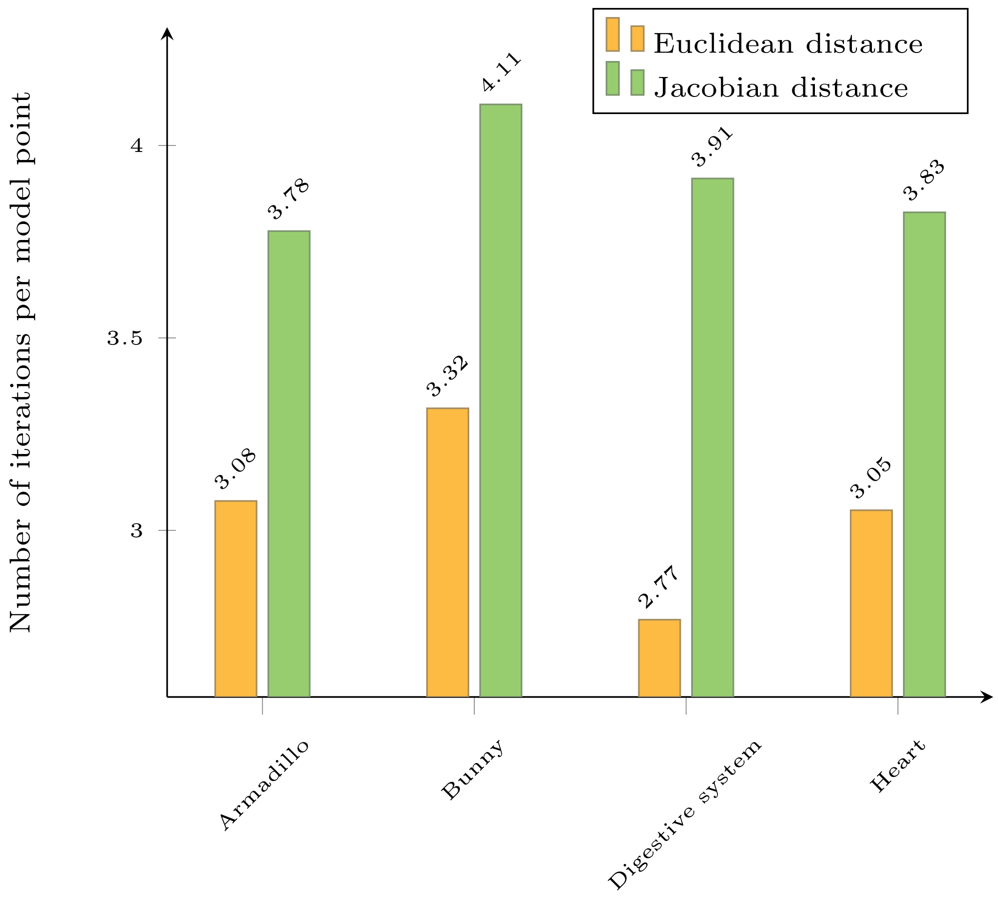


Figure 4.6: Average number of iterations per model point that the Newton Method needs to converge on a good set of T-Celestial Coordinates. The first bar of each group shows the number of iterations when the method is initialised with the help of an *euclidean* neighbour. The second bar refers to the number of iterations when the method uses a *jacobian* neighbour.

convex hull, as shown in Figure 4.7. Moreover, this system of coordinates does not have topology awareness of the geometry of the model, hence the deformations are not shape-aware. The second disadvantage is that they are not rotational-invariant.

4.4 S-Celestial Coordinates

In the previous section, we have presented the derivation of the T-Celestial Coordinates and also their main disadvantages; one being the fact that they are not rotational-invariant. Hence, in an effort to design a Point-Based Celestial Coordinate system that met the rotational-invariant property, we developed the S-Celestial Coordinates.

Given a model point $\mathbf{p}$ and a user-defined constant $\lambda \geq 1$, the method defines the function

$$f(\mathbf{c}) = \frac{1}{2\sum_{j=1}^{n} \mu_j} \sum_{i=1}^{n} \mu_i^2 \vec{\mathbf{q}}_i. \tag{4.10}$$

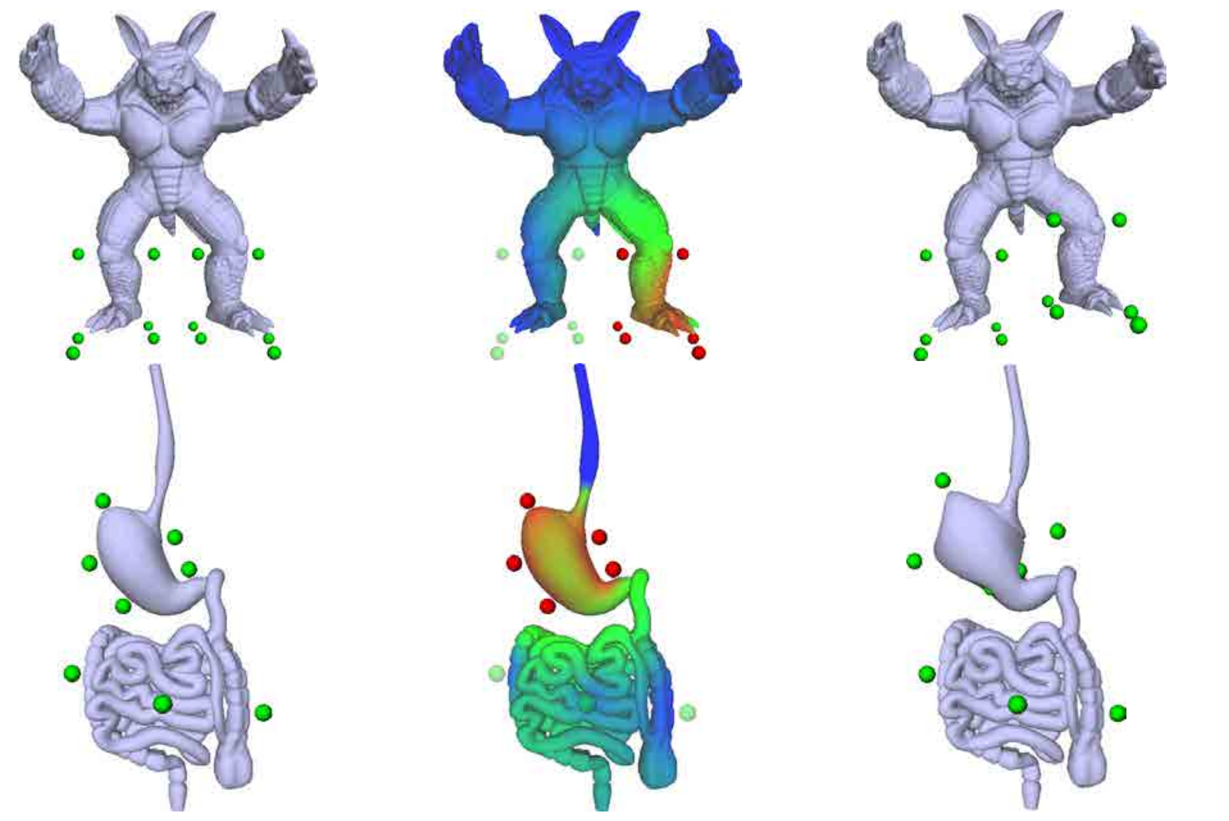


Figure 4.7: Example of two deformation processes driven by the T-Celestial Coordinates. (a) shows the initial configuration of the *Armadillo* and the *Digestive system* models surrounded by deformation handles. Both models are enclosed in a cube-shaped convex hull, however, it can not be seen due to the zoom of the images. (b) presents the heatmap corresponding to the control points in red. The global effect of the coordinates can especially be seen in the *Digestive system*. Observe that, although the red control points wrap the stomach, almost all the bowels are also affected by them. Finally, (c) shows the result of the deformation driven by the red handles in (b).

This function is the center of mass $\boldsymbol{\ell} = f(\mathbf{c})$ of the segments defined by the intersection of the rays emanating from the point $\mathbf{p}$, that have the direction of the projected control points $\vec{\mathbf{q}}_i$, with the sphere $\mathcal{S}_\mathbf{c}$. This sphere is centered at $\mathbf{c}$ and its radius is $r = \lambda + \|(\mathbf{c}-\mathbf{p})\|$, hence, it completely wraps the unit sphere $\mathcal{S}_\mathbf{p}$ centered at $\mathbf{p}$, as shown in Figure 4.8. We name $\mu_i\vec{\mathbf{q}}_i$ the intersection points between the rays and the sphere $\mathcal{S}_\mathbf{c}$. Then, the center of the mass is evaluated on the mid-points $\frac{\mu_i}{2}\vec{\mathbf{q}}_i$, whose associated mass is equal to the length μ_i of the corresponding segment.

The technique, therefore, consists in looking for a correct point $\mathbf{c}$ such that the center of the mass $\boldsymbol{\ell}$ of the intersected segments coincides with the model point $\mathbf{p}$. That is

$$\boldsymbol{\ell} = \frac{1}{2\sum_{j=1}^{n}\mu_j}\sum_{i=1}^{n}\mu_i^2\vec{\mathbf{q}}_i = \mathbf{0},$$

where $\mu_i \geq 0$ and $\mu_i\vec{\mathbf{q}}_i \in \mathcal{S}_\mathbf{c}$. Once this goal is met, the β_i coordinates are those shown below

$$\beta_i = \frac{\mu_i^2}{2\sum_{j=1}^{n}\mu_j}.$$

Lastly, the final coordinates α_i are computed as done in [FKR05] and shown in Equation (4.2).

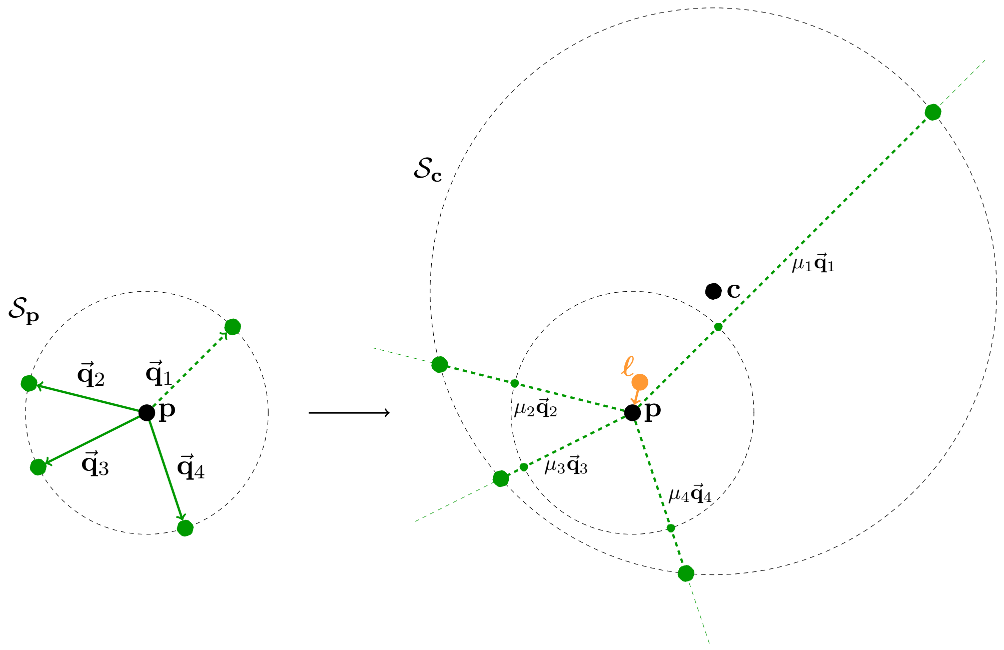


Figure 4.8: The unit sphere $\mathcal{S}_\mathbf{p}$ and the projected control points $\vec{\mathbf{q}}_i$ are shown on the left part of the image. In contrast, the right side shows a random wrapping sphere $\mathcal{S}_\mathbf{c}$, the intersected segments and their center of mass $\boldsymbol{\ell}$. We want to find that sphere $\mathcal{S}_\mathbf{c}$ such that $\boldsymbol{\ell}$ equals $\mathbf{p}$.

Following these lines, we detail the mathematical expression of the intersection points $\mu_i\vec{\mathbf{q}}_i$ in Section 4.4.1. Then, we demonstrate the existence of the proposed S-Celestial Coordinates in Section 4.4.2 and describe the methodology to numerically compute them in Section 4.4.3. Finally, Section 4.4.4 analyses the results obtained.

4.4.1 The Ray-Sphere Intersection

This section is devoted to deriving the mathematical expression to compute the intersection points between the rays that emanate from the point $\mathbf{p}$ and have the direction of the projected control points $\vec{\mathbf{q}}_i$ and the sphere $\mathcal{S}_\mathbf{c}$. Without losing generality and for the sake of simplicity, let us assume that the point $\mathbf{p}$ is located at the origin of the euclidean space, that is $\mathbf{p} = (0,0,0)$.

On the one hand, we name r_i the ray that corresponds to the projected control point $\vec{\mathbf{q}}_i$. It can be defined as

$$r_i = t\vec{\mathbf{q}}_i, \qquad t > 0. \tag{4.11}$$

On the other hand, the sphere $\mathcal{S}_\mathbf{c}$ centered at point $\mathbf{c}$ and with radius $r = \lambda + \|(\mathbf{c}-\mathbf{p})\| = \lambda + \|\mathbf{c}\|$ is described as follows

$$\mathcal{S}_\mathbf{c} : (x-c_x)^2 + (y-c_y)^2 + (z-c_z)^2 = r^2 \tag{4.12}$$

Additionally, we know that the ray r_i intersects the sphere $\mathcal{S}_\mathbf{c}$ when $t = \mu_i$. Therefore, we substitute the value μ_i into Equation (4.11) and this equation into the Expression (4.12) to finally compute the intersection

$$\begin{aligned} (r_i - \mathbf{c})^2 &= r^2 \\ (\mu_i \vec{\mathbf{q}}_i - \mathbf{c})^2 &= (\lambda + \|\mathbf{c}\|)^2. \end{aligned} \tag{4.13}$$

Now, we can expand Equation (4.13)

$$\begin{aligned} &\mu_i^2 \vec{\mathbf{q}}_i^2 - 2\mu_i \vec{\mathbf{q}}_i \cdot \mathbf{c} + \mathbf{c}^2 = \lambda^2 + 2\lambda\|\mathbf{c}\| + \|\mathbf{c}\|^2 \\ &\mu_i^2 \|\vec{\mathbf{q}}_i\| - 2\mu_i \vec{\mathbf{q}}_i \cdot \mathbf{c} + \|\mathbf{c}\|^2 = \lambda^2 + 2\lambda\|\mathbf{c}\| + \|\mathbf{c}\|^2 \\ &\mu_i^2 - 2\mu_i \vec{\mathbf{q}}_i \cdot \mathbf{c} - \lambda^2 - 2\lambda\|\mathbf{c}\| = 0 \end{aligned}$$

and isolate μ_i

$$\begin{aligned} \mu_i &= \frac{2\vec{\mathbf{q}}_i \cdot \mathbf{c} + \sqrt{(2\vec{\mathbf{q}}_i \cdot \mathbf{c})^2 + 4(\lambda^2 + 2\lambda\|\mathbf{c}\|)}}{2} \\ \mu_i &= \vec{\mathbf{q}}_i \cdot \mathbf{c} + \sqrt{(\vec{\mathbf{q}}_i \cdot \mathbf{c})^2 + \lambda^2 + 2\lambda\|\mathbf{c}\|} \end{aligned} \tag{4.14}$$

Notice that $\sqrt{(\vec{\mathbf{q}}_i \cdot \mathbf{c})^2 + \lambda^2 + 2\lambda\|\mathbf{c}\|} > \vec{\mathbf{q}}_i \cdot \mathbf{c}$, hence we always take the positive value of the square root to ensure the positivity of μ_i, which is required by the definition of the rays in Equation (4.11).

Once we have obtained the expression for the weights μ_i, we can substitute it in the original Function (4.10). Thus,

$$f(\mathbf{c}) = \frac{\sum_{i=1}^{n} \left(\vec{\mathbf{q}}_i \cdot \mathbf{c} + \sqrt{(\vec{\mathbf{q}}_i \cdot \mathbf{c})^2 + \lambda^2 + 2\lambda\|\mathbf{c}\|} \right)^2 \vec{\mathbf{q}}_i}{2\sum_{j=1}^{n} \vec{\mathbf{q}}_j \cdot \mathbf{c} + \sqrt{(\vec{\mathbf{q}}_j \cdot \mathbf{c})^2 + \lambda^2 + 2\lambda\|\mathbf{c}\|}}. \tag{4.15}$$

4.4.2 Existence of the S-Celestial Coordinates

In this section we want to demonstrate the existence of the S-Celestial Coordinates. In other words, we want to prove that, for any model point $\mathbf{p}$, there exists a point $\mathbf{c}$ such that

$$f(\mathbf{c}) = \frac{1}{2\sum_{j=1}^{n} \mu_j} \sum_{i=1}^{n} \mu_i^2 \vec{\mathbf{q}}_i = \mathbf{0}.$$

Thus, we can compute a correct set of S-Celestial Coordinates for $\mathbf{p}$.

Consider the function

$$g(\mathbf{c}) = \mathbf{c} - f(\mathbf{c}). \tag{4.16}$$

The idea is to look for a convex and compact domain $\mathcal{B}$ such that, if we limit the function $g(\mathbf{c})$ to it, its images also belong to the same domain

$$\begin{aligned} g: \ &\mathcal{B} \ \rightarrow \ \mathcal{B} \\ &\mathbf{c} \ \rightarrow \ g(\mathbf{c}). \end{aligned}$$

Once we have established $\mathcal{B}$, we can apply the Brouwer Fixed Point Theorem [Bro11], which states that there exists a point $\mathbf{c}' \in \mathcal{B}$ such that $g(\mathbf{c}') = \mathbf{c}'$. Therefore, $f(\mathbf{c}') = \mathbf{c}' - g(\mathbf{c}') = \mathbf{0}$.

Still assuming $\mathbf{p}$ is at the origin and for any given point $\mathbf{c}$, let us define the vector $\vec{\mathbf{e}} = \frac{\mathbf{c}-\mathbf{p}}{\|\mathbf{c}-\mathbf{p}\|}$ such that $\mathbf{c} = \gamma\vec{\mathbf{e}}$ and $\gamma = \|\mathbf{c} - \mathbf{p}\|$.

Lemma 1. *We define the ball* $\mathcal{B}^1_{\mathbf{c}} = \{\mathbf{x} : d(\mathbf{x}, \frac{\mathbf{c}}{2}) \leq \frac{\gamma+\lambda}{2}\}$ *centered on* $\frac{\mathbf{c}}{2}$ *and with radius* $\frac{\gamma+\lambda}{2}$. *Then,* $f(\mathbf{c}) \in \mathcal{B}^1_{\mathbf{c}}$.

Proof. We know that the points $\mu_i \vec{\mathbf{q}}_i \in \mathcal{S}_{\mathbf{c}}$. Then, $\frac{\mu_i}{2}\vec{\mathbf{q}}_i \in \mathcal{S}^1_{\mathbf{c}}$, where $\mathcal{S}^1_{\mathbf{c}}$ is the boundary of the ball $\mathcal{B}^1_{\mathbf{c}}$. We only need to notice that $f(\mathbf{c})$ is a convex combination of the points $\frac{\mu_i}{2}\vec{\mathbf{q}}_i$ (see Equation (4.10)). This means that $f(\mathbf{c})$ is in the convex hull of these points, $ch(\{\frac{\mu_i}{2}\vec{\mathbf{q}}_i\})$. Additionally, the points $\frac{\mu_i}{2}\vec{\mathbf{q}}_i$ are on the sphere $\mathcal{S}^1_{\mathbf{c}}$, so their convex hull is inscribed in the ball $\mathcal{B}^1_{\mathbf{c}}$. Consequently,

$$f(\mathbf{c}) \in ch\left(\left\{\frac{\mu_i}{2}\vec{\mathbf{q}}_i\right\}\right) \in \mathcal{B}^1_{\mathbf{c}},$$

which is what we set out to demonstrate. □

Lemma 2. *Let* $\mathcal{B}^2_{\mathbf{c}} = \{\mathbf{x} : d(\mathbf{x}, \mathbf{c}) \leq \gamma\}$ *be the ball centered on* $\mathbf{c}$ *and with radius* γ. *Then, the intersection between the balls* $\mathcal{B}^1_{\mathbf{c}}$ *and* $\mathcal{B}^2_{\mathbf{c}}$ *is not empty. That is,* $\mathcal{B}^1_{\mathbf{c}} \cap \mathcal{B}^2_{\mathbf{c}} \neq \emptyset$.

Proof. Notice that $\|\mathbf{c} - \frac{\mathbf{c}}{2}\| = \frac{\gamma}{2}$, which means that $\mathbf{c}$ lies inside $\mathcal{B}^1_{\mathbf{c}}$ and $\frac{\mathbf{c}}{2}$ lies inside $\mathcal{B}^2_{\mathbf{c}}$. Therefore, $\mathcal{B}^1_{\mathbf{c}} \cap \mathcal{B}^2_{\mathbf{c}} \neq \emptyset$ as we wanted to demostrate. □

Next, we develop three Propositions to demonstrate that the compact domain $\mathcal{B}$ exists. Hence, we can assert that the S-Celestial Coordinates are correct.

Proposition 1. *For any vector* $\vec{\mathbf{e}}$, *there exists a value* $\bar{\gamma}$ *such that, for any* $\gamma \geq \bar{\gamma}$, $f(\mathbf{c}) \in \mathcal{B}^1_{\mathbf{c}} \cap \mathcal{B}^2_{\mathbf{c}}$.

Proof. For the sake of simplicity, let us fix the vector $\vec{\mathbf{e}} = (1, 0, 0)$. Observe that the system is rotational-invariant and, therefore, any arbitrary vector $\vec{\mathbf{e}}'$ can be rotated such that

$$R' \cdot \vec{\mathbf{e}}' = \vec{\mathbf{e}} = (1, 0, 0).$$

The rotation matrix R' can also be applied to the vectors $\vec{\mathbf{q}}_i$ and the demonstration remains valid.

We know that $\mathbf{c} = \gamma\vec{\mathbf{e}}$, so $\mathbf{c} = (\gamma, 0, 0)$. The balls $\mathcal{B}^1_{\mathbf{c}}$ and $\mathcal{B}^2_{\mathbf{c}}$ are those defined in Lemmas 1 and 2, respectively. Then, $\mathcal{S}^1_{\mathbf{c}}$ and $\mathcal{S}^2_{\mathbf{c}}$ are the spheres that correspond to these balls. Finally, $\mathbf{a}$ represents an intersection point between these spheres. This geometry is represented in Figure 4.9.

From Lemma 1, we know that $f(\mathbf{c}) \in \mathcal{B}^1_{\mathbf{c}}$ always, so we only need to demonstrate that, for a larger enough value γ, $f(\mathbf{c}) \in \mathcal{B}^2_{\mathbf{c}}$. However, we go a little further and we prove that $[f(\mathbf{c})]_x \geq a_x$. In other words, we demonstrate that $f(\mathbf{c})$ lies on the right of the plane defined by the intersection of both spheres. This plane is represented by the blue line in Figure 4.9.

First, we study the intersection points $\mathbf{a}$ and present the equation which defines them and, particularly, their components a_x. On the one hand, we know that $\mathbf{a}$ belongs to the sphere $\mathcal{S}^1_{\mathbf{c}}$, the boundary of the ball $\mathcal{B}^1_{\mathbf{c}}$, hence

$$\begin{aligned} \left(\mathbf{a} - \frac{\mathbf{c}}{2}\right)^2 &= \left(\frac{\gamma+\lambda}{2}\right)^2 \\ \left(a_x - \frac{\gamma}{2}\right)^2 + a_y^2 + a_z^2 &= \left(\frac{\gamma+\lambda}{2}\right)^2 \qquad (4.17) \\ \mathbf{a}^2 - a_x\gamma - \frac{2\gamma\lambda + \lambda^2}{4} &= 0. \end{aligned}$$

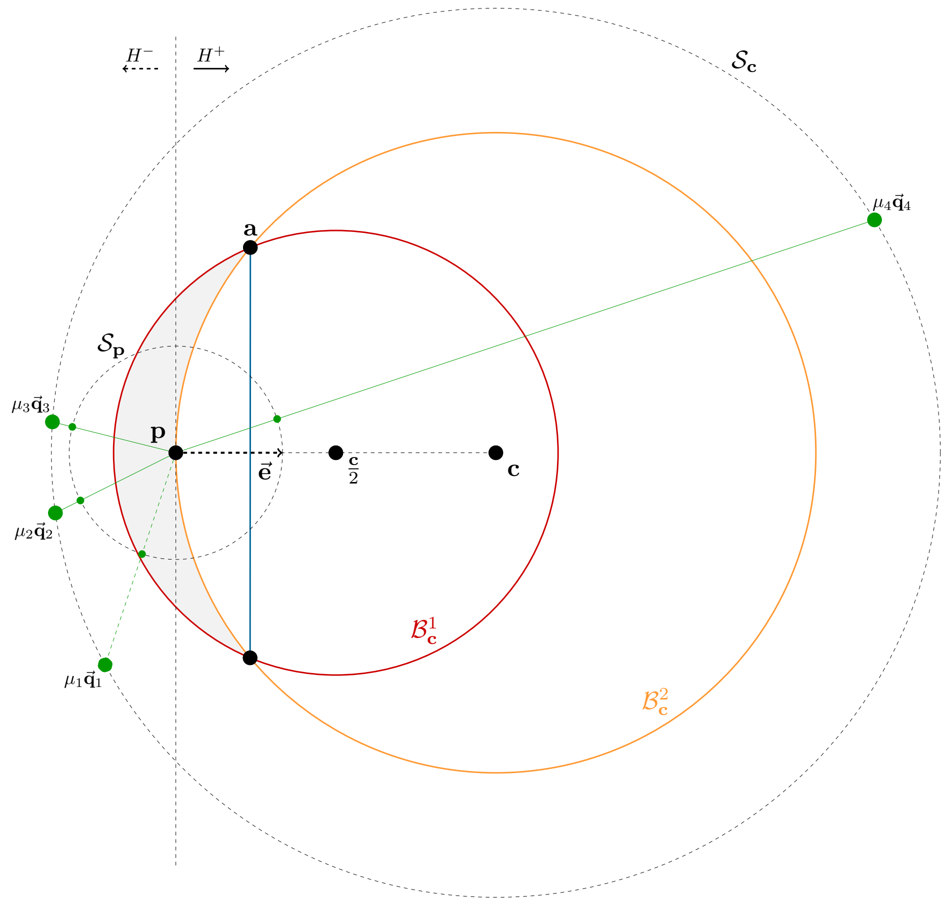

H^-
H^+
$\mathcal{S}_\mathbf{c}$
$\mu_4\vec{\mathbf{q}}_4$
$\mathbf{a}$
$\mathcal{S}_\mathbf{p}$
$\mu_3\vec{\mathbf{q}}_3$
$\mathbf{p}$
$\vec{\mathbf{e}}$
$\frac{\mathbf{c}}{2}$
$\mathbf{c}$
$\mu_2\vec{\mathbf{q}}_2$
$\mathcal{B}^1_\mathbf{c}$
$\mu_1\vec{\mathbf{q}}_1$
$\mathcal{B}^2_\mathbf{c}$

Figure 4.9: Representation of the geometry that involves the unit sphere $\mathcal{S}_\mathbf{p}$, centered at the model point $\mathbf{p}$, and the balls $\mathcal{B}^1_\mathbf{c}$ and $\mathcal{B}^2_\mathbf{c}$.

On the other hand, it also belongs to the sphere $\mathcal{S}^2_\mathbf{c}$, the boundary of $\mathcal{B}^2_\mathbf{c}$

$$\begin{aligned}(\mathbf{a}-\mathbf{c})^2 &= \gamma^2\\ (a_x-\gamma)^2+a_y^2+a_z^2 &= \gamma^2\\ \mathbf{a}^2-2a_x\gamma &= 0.\end{aligned} \tag{4.18}$$

Now, we can equate Equations (4.17) and (4.18) and isolate the component a_x

$$\begin{aligned}\mathbf{a}^2-a_x\gamma-\frac{2\gamma\lambda+\lambda^2}{4} &= \mathbf{a}^2-2a_x\gamma\\ a_x\gamma-\frac{2\gamma\lambda+\lambda^2}{4} &= 0\\ a_x &= \frac{2\gamma\lambda+\lambda^2}{4\gamma}\\ a_x &= \frac{\lambda}{2}+\frac{\lambda^2}{4\gamma}.\end{aligned}$$

Notice that the component $\frac{\lambda}{2}$ is constant while the component $\frac{\lambda^2}{4\gamma}$ becomes smaller as γ grows. Therefore, we can conclude that the component a_x decreases and tends to $\frac{\lambda}{2}$ whenever γ increases.

Now, we study the behaviour of the function $f(\mathbf{c})$. A basic assumption of the present technique is that the model point $\mathbf{p}$ lies inside the convex hull of the control points. Consequently, it also lies in the convex hull of their projections $\vec{\mathbf{q}}_{\mathbf{i}}$ over its unit sphere $\mathcal{S}_{\mathbf{p}}$. This means that in any hemisphere there is a point. See the example in Figure 4.9.

Based on this fact and on Equations (4.10) and (4.15), we want to analyse the evolution of the component x of the function $f(\mathbf{c})$ depending on the value γ. Thus, let us assume that the first ℓ control points are those on the negative hemisphere while the last $m = n - \ell$ lie on the positive hemisphere. Then, we can split Equation (4.10) as follows

$$f(\mathbf{c}) = f(\gamma) = \frac{\sum_{i=1}^{\ell} \mu_i^2 \vec{\mathbf{q}}_i}{2\sum_{k=1}^{n} \mu_k} + \frac{\sum_{j=\ell+1}^{n} \mu_j^2 \vec{\mathbf{q}}_j}{2\sum_{k=1}^{n} \mu_k}. \tag{4.19}$$

Hence, the component $[f(\mathbf{c})]_x$ is

$$[f(\mathbf{c})]_x = [f(\gamma)]_x = \frac{\sum_{i=1}^{\ell} \mu_i^2 q_{i_x}}{2\sum_{k=1}^{n} \mu_k} + \frac{\sum_{j=\ell+1}^{n} \mu_j^2 q_{j_x}}{2\sum_{k=1}^{n} \mu_k}.$$

The minimum of the sum $\frac{\sum_{i=1}^{\ell} \mu_i^2 q_{i_x}}{2\sum_{k=1}^{n} \mu_k}$ is bound by the value $-\frac{\lambda}{2}$. Therefore,

$$[f(\mathbf{c})]_x > -\frac{\lambda}{2} + \frac{\sum_{j=\ell+1}^{n} \mu_j^2 q_{j_x}}{2\sum_{k=1}^{n} \mu_k}. \tag{4.20}$$

From Equation (4.14) we know that $\mu_i = \vec{\mathbf{q}}_i\mathbf{c} + \sqrt{(\vec{\mathbf{q}}_i\mathbf{c})^2 + \lambda^2 + 2\lambda\|\mathbf{c}\|}$. Hence, we can substitute the point $\mathbf{c} = (\gamma, 0, 0)$, so the new expression for the weights μ_i is $\mu_i = \gamma q_{i_x} + \sqrt{(\gamma q_{i_x})^2 + \lambda^2 + 2\lambda\gamma} > 0$. We replace it into Equation (4.20), so

$$[f(\mathbf{c})]_x > -\frac{\lambda}{2} + \frac{\sum_{j=\ell+1}^{n} \left(\gamma q_{j_x} + \sqrt{(\gamma q_{j_x})^2 + \lambda^2 + 2\lambda\gamma}\right)^2 q_{j_x}}{2\sum_{k=1}^{n} \gamma q_{k_x} + \sqrt{(\gamma q_{k_x})^2 + \lambda^2 + 2\lambda\gamma}}.$$

Since $q_{j_x} \in (0, 1]$, the sum $\frac{\sum_{j=\ell+1}^{n} \left(\gamma q_{j_x} + \sqrt{(\gamma q_{j_x})^2 + \lambda^2 + 2\lambda\gamma}\right)^2 q_{j_x}}{2\sum_{k=1}^{n} \gamma q_{k_x} + \sqrt{(\gamma q_{k_x})^2 + \lambda^2 + 2\lambda\gamma}}$ is always positive. Additionally, we can see that the numerator is squared, which means that it grows faster than the denominator when γ also grows. This allows us to assert that the component $[f(\mathbf{c})]_x = [f(\gamma)]_x$ increases without bound as the value γ increases.

Finally, we have demonstrated that the component a_x decreases and tends to $\frac{\lambda}{2}$ whenever γ increases. We have also demonstrated that the component $[f(\mathbf{c})]_x$ increases without bound as the value γ also increases. That means that, at a certain value γ, the component $[f(\mathbf{c})]_x$ overtakes the component a_x, which implies that the function $f(\mathbf{c})$ lies inside the ball $\mathcal{B}_{\mathbf{c}}^2$ when γ is big enough. Therefore, we can assert that there exists a bounding value $\bar{\gamma}$ such that, $\forall \gamma \geq \bar{\gamma}$, then, $f(\mathbf{c}) \in \mathcal{B}_{\mathbf{c}}^1 \cap \mathcal{B}_{\mathbf{c}}^2$, as we wanted to demonstrate. □

Proposition 2. *For any vector* $\vec{\mathbf{e}}$*, there exists a value* $\bar{\bar{\gamma}} = \max_{\vec{\mathbf{e}}} \bar{\gamma}(\vec{\mathbf{e}})$ *such that,* $\forall \gamma > \bar{\bar{\gamma}}$*, then* $g : \partial\mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}} \to \mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}$*. The ball* $\mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}$ *is defined as* $\mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}} = \{\mathbf{x} : d(\mathbf{x}, \mathbf{p}) \leq \bar{\bar{\gamma}}\}$.

Proof. From Proposition 1, we know that for any vector $\vec{\mathbf{e}}$ there exists a bounding value $\bar{\gamma}$ such that $\forall \gamma \geq \bar{\gamma}$, then, $f(\mathbf{c}) \in \mathcal{B}_{\mathbf{c}}^1 \cap \mathcal{B}_{\mathbf{c}}^2$. Now, let us name $\bar{\bar{\gamma}} = \max_{\vec{\mathbf{e}}} \bar{\gamma}(\vec{\mathbf{e}})$, the maximum bounding value $\bar{\gamma}$ among all the vectors $\vec{\mathbf{e}}$. Since, $\bar{\bar{\gamma}} \geq \bar{\gamma}$ for any $\bar{\gamma}$ and $\vec{\mathbf{e}}$, by Proposition 1, $f(\mathbf{c}) \in \mathcal{B}_{\mathbf{c}}^1 \cap \mathcal{B}_{\mathbf{c}}^2$, where the balls $\mathcal{B}_{\mathbf{c}}^1$ and $\mathcal{B}_{\mathbf{c}}^2$ are defined from the value $\bar{\bar{\gamma}}$.

From Equation (4.16), the function $g(\mathbf{c}) = \mathbf{c} - f(\mathbf{c})$. Hence, its magnitude is $\|g(\mathbf{c})\| = \|\mathbf{c} - f(\mathbf{c})\|$. Observe that $f(\mathbf{c}) \in \mathcal{B}^1_{\mathbf{c}} \cap \mathcal{B}^2_{\mathbf{c}}$ which means that, particularly, $f(\mathbf{c}) \in \mathcal{B}^2_{\mathbf{c}}$. By definition, $\forall f(\mathbf{c}) \in \mathcal{B}^2_{\mathbf{c}}$, $\|\mathbf{c} - f(\mathbf{c})\| \leq \bar{\bar{\gamma}}$, hence $\|g(\mathbf{c})\| \leq \bar{\bar{\gamma}}$, which proves that $g(\mathbf{c}) \in \mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}$, as we wanted to demonstrate. □

Proposition 3. *Let* $\bar{\bar{\bar{\gamma}}} = \max_{\mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}} \|g(\mathbf{c})\|$ *and* $\mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}} \leq \{\mathbf{x} : d(\mathbf{x},\mathbf{p}) = \bar{\bar{\bar{\gamma}}}\}$. *Then* $\bar{\bar{\bar{\gamma}}}$ *ensures that* $g : \mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}} \to \mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}}$. *Therefore, there exists a solution* $f(\mathbf{c}) = \mathbf{0}$ *for* $\|\mathbf{c} - \mathbf{p}\| \leq \bar{\bar{\bar{\gamma}}}$.

Proof. We must deal with two different situations. On the one hand, $\mathbf{c} \in \mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}} - \mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}$ whose proper behaviour has been demonstrated in Proposition 2. On the other, $\mathbf{c} \in \mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}$.

Notice that we define $\bar{\bar{\bar{\gamma}}}$ as the maximum magnitude that the function $g(\mathbf{c})$ can reach for any point $\mathbf{c} \in \mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}$. Hence, the ball $\mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}}$ also includes $g(\mathbf{c})$ when $\mathbf{c} \in \mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}$. Consequently, $\bar{\bar{\bar{\gamma}}}$ is the minimum radius that ensures

$$g : \mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}} \to \mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}}$$

and, therefore, there exists a solution $f(\mathbf{c}) = \mathbf{0}$ for $\|\mathbf{c} - \mathbf{p}\| = \bar{\bar{\bar{\gamma}}}$ as we wanted to demonstrate.

In addition, we can easily compute the lower bound for $\bar{\bar{\bar{\gamma}}}$. Figure 4.10 illustrates a situation where $f(\mathbf{c}) \notin \mathcal{B}^1_{\mathbf{c}} \cap \mathcal{B}^2_{\mathbf{c}}$. In the worst case, $f(\mathbf{c}) \in \partial\mathcal{B}^1_{\mathbf{c}}$ lies in the opposite direction to $\mathbf{c}$. Then, the maximum value for $\|g(\mathbf{c})\|$ is $\frac{2(\bar{\bar{\gamma}}-\epsilon)+\lambda}{2}$, so we can define $\bar{\bar{\bar{\gamma}}} \geq \left(\bar{\bar{\gamma}} + \frac{\lambda}{2}\right)$.

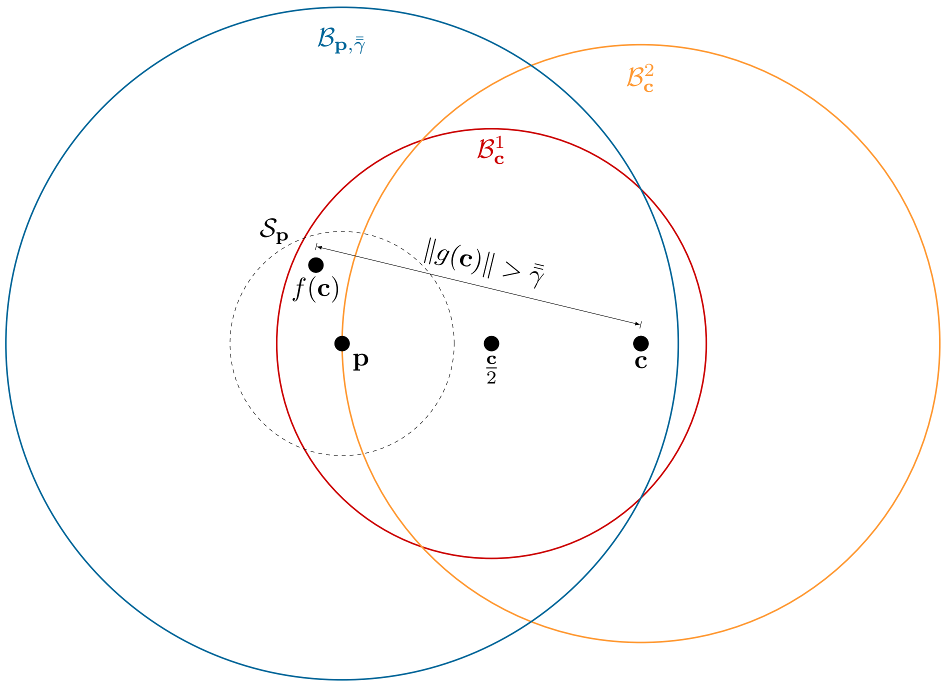

$\mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}$
$\mathcal{B}^2_{\mathbf{c}}$
$\mathcal{B}^1_{\mathbf{c}}$
$\mathcal{S}_{\mathbf{p}}$
$\|g(\mathbf{c})\| > \bar{\bar{\gamma}}$
$f(\mathbf{c})$
$\mathbf{p}$
$\frac{\mathbf{c}}{2}$
$\mathbf{c}$

Figure 4.10: Conflict situation that can occur for a point $\mathbf{c} \in \mathcal{B}_{\mathbf{p},\bar{\bar{\gamma}}}$.

□

To sum up, we have demonstrated that there exists a value $\bar{\bar{\bar{\gamma}}}$ that ensures $g : \mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}} \to \mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}}$. Hence, we define the compact domain $\mathcal{B}$ for the function $g(\mathbf{c})$ as $\mathcal{B} = \mathcal{B}_{\mathbf{p},\bar{\bar{\bar{\gamma}}}}$ and declare that the S-Celestial Coordinates can be correctly computed for any model point $\mathbf{p}$.

4.4.3 Solving for $f(\mathbf{c})$

In Section 4.4.1, we have presented the complete formula to compute the intersection points $\mu_i\vec{\mathbf{q}}_i$ and, hence, the function $f(\mathbf{c})$. Section 4.4.2 proves that there is always a solution for it. Therefore, we can now present the methodology to solve this function

$$f(\mathbf{c}) = \frac{\sum_{i=1}^{n}\left(\vec{\mathbf{q}}_i\cdot\mathbf{c} + \sqrt{(\vec{\mathbf{q}}_i\cdot\mathbf{c})^2+\lambda^2+2\lambda\|\mathbf{c}\|}\right)^2\vec{\mathbf{q}}_i}{2\sum_{j=1}^{n}\vec{\mathbf{q}}_j\cdot\mathbf{c} + \sqrt{(\vec{\mathbf{q}}_j\cdot\mathbf{c})^2+\lambda^2+2\lambda\|\mathbf{c}\|}} = \mathbf{0}, \tag{4.21}$$

as exposed in Section 4.4.1.

Notice that Function (4.21) can be simplified into the following one

$$f(\mathbf{c}) = \sum_{i=1}^{n}\left(\vec{\mathbf{q}}_i\cdot\mathbf{c} + \sqrt{(\vec{\mathbf{q}}_i\cdot\mathbf{c})^2+\lambda^2+2\lambda\|\mathbf{c}\|}\right)^2\vec{\mathbf{q}}_i = \mathbf{0}, \tag{4.22}$$

which is a nonlinear system of equations. As a consequence, we also propose using the Newton Method [Hil56] to solve it, as done to compute the solution for the System (4.7) in the case of the T-Celestial Coordinates, see Section 4.3.2. In the particular situation of the S-Celestial Coordinates, each iteration k of the Newton Method consists in solving the matrix system below

$$\mathbf{c}^k = \mathbf{c}^{k-1} - \left(J(f(\mathbf{c}^{k-1}))^{-1}\right)^{\top}\cdot f(\mathbf{c}^{k-1}),$$

where $\mathbf{c}^{k-1}$ is the solution of the previous iteration and J is the jacobian matrix of the System (4.22). This jacobian matrix is

$$J(f(\mathbf{c})) = 2\begin{pmatrix}\sum_{i=1}^{n} t_{i_x}q_{i_x} & \sum_{i=1}^{n} t_{i_y}q_{i_x} & \sum_{i=1}^{n} t_{i_z}q_{i_x}\\ \sum_{i=1}^{n} t_{i_x}q_{i_y} & \sum_{i=1}^{n} t_{i_y}q_{i_y} & \sum_{i=1}^{n} t_{i_z}q_{i_y}\\ \sum_{i=1}^{n} t_{i_x}q_{i_z} & \sum_{i=1}^{n} t_{i_y}q_{i_z} & \sum_{i=1}^{n} t_{i_z}q_{i_z}\end{pmatrix}, \tag{4.23}$$

where

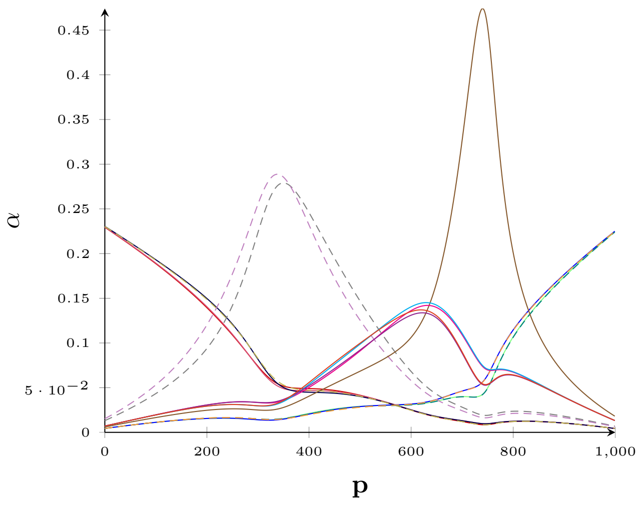
$$\begin{aligned} t_{i_x} &= \left(\mathbf{c}\cdot\vec{\mathbf{q}}_i + \sqrt{(\vec{\mathbf{q}}_i\cdot\mathbf{c})^2+\lambda^2+2\lambda\|\mathbf{c}\|}\right)\left(q_{i_x} + \frac{\mathbf{c}\cdot\vec{\mathbf{q}}_i q_{i_x} + \frac{c_x\lambda}{\|\mathbf{c}\|}}{\sqrt{(\vec{\mathbf{q}}_i\cdot\mathbf{c})^2+\lambda^2+2\lambda\|\mathbf{c}\|}}\right)\\ t_{i_y} &= \left(\mathbf{c}\cdot\vec{\mathbf{q}}_i + \sqrt{(\vec{\mathbf{q}}_i\cdot\mathbf{c})^2+\lambda^2+2\lambda\|\mathbf{c}\|}\right)\left(q_{i_y} + \frac{\mathbf{c}\cdot\vec{\mathbf{q}}_i q_{i_y} + \frac{c_y\lambda}{\|\mathbf{c}\|}}{\sqrt{(\vec{\mathbf{q}}_i\cdot\mathbf{c})^2+\lambda^2+2\lambda\|\mathbf{c}\|}}\right)\\ t_{i_z} &= \left(\mathbf{c}\cdot\vec{\mathbf{q}}_i + \sqrt{(\vec{\mathbf{q}}_i\cdot\mathbf{c})^2+\lambda^2+2\lambda\|\mathbf{c}\|}\right)\left(q_{i_z} + \frac{\mathbf{c}\cdot\vec{\mathbf{q}}_i q_{i_z} + \frac{c_z\lambda}{\|\mathbf{c}\|}}{\sqrt{(\vec{\mathbf{q}}_i\cdot\mathbf{c})^2+\lambda^2+2\lambda\|\mathbf{c}\|}}\right). \end{aligned} \tag{4.24}$$

Just as we have exposed in the case of the T-Celestial Coordinates, we also need to select a first approximation of the solution $\mathbf{c}^0$ and to define an ending condition that ensures a sufficiently accurate result. On the one hand, we consider a precise enough solution when the function $f(\mathbf{c})$, as defined in Expression (4.21), reaches $f(\mathbf{c}) \leq \epsilon$, for a proposed value $\epsilon = 10^{-6}$. On the other hand, we suggest a similar procedure as that used for the T-Celestial Coordinates to accelerate the Newton Method by an initial guess. First, we take an initial model point $\mathbf{p}_0$, the seed point, and compute its set of S-Celestial Coordinates. Once the seed point $\mathbf{p}_0$ has its own coordinates defined, the final solution for $\mathbf{c}_{\mathbf{p}_0} = (c_x, c_y, c_z)$ is spread among its neighbours[3], so they can use it as their initial guess to compute their own coordinates.

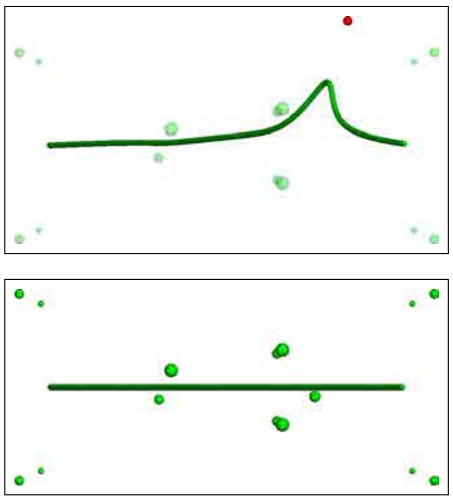
We define the initial approximation $\mathbf{c}^0$ for the Newton Method in $\mathbf{p}_0$ as the inverted center of mass $\mathbf{c}^0 = -\boldsymbol{\ell}$ of the intersected segments $\mu_i \vec{\mathbf{q}}_i$ with a sphere centered on $\mathbf{p}$ with a radius $r = \lambda$. Concerning the method to spread the final solution $\mathbf{c}_\mathbf{p}$ among the neighbours of a certain model point $\mathbf{p}$, we use the same algorithm presented in the case of the T-Celestial Coordinates to sort and queue these neighbouring points. However, in the case of the S-Celestial Coordinates we only use the *euclidean distance* $d_E = \|\mathbf{p} - \mathbf{p}_j\|$ to compute the closest neighbour to a given point $\mathbf{p}$.

4.4.4 Discussion

The S-Celestial Coordinates, as the T-Celestial Coordinates in Section 4.3, are a well-defined system of Point-Based Celestial Coordinates. They are efficient, positive, continuous and smooth, as shown in Figure 4.11. Moreover, if we compare Figure 4.4 with Figure 4.11 we can also observe that both systems of coordinates are extremely similar. However, the S-Celestial Coordinates, in contrast with the T-Celestial Coordinates approach, are rotational-invariant.



(a)



(b)

Figure 4.11: The S-Celestial Coordinate functions corresponding to the initial configuration of a line surrounded by fifteen control points are shown in (a). Additionally, the bottom and the top images in (b) show this initial configuration and a simple deformation produced by the translation of the control point in red, respectively.

Regarding the efficiency of their computation, we have compared the computational cost of computing the T-Celestial Coordinates and the S-Celestial Coordinates. As shown in Figure 4.12, the S-Celestial Coordinates are much more efficient than the T-Celestial Coordinates[4].

[3]As in the case of the T-Celestial Coordinates, two points $\mathbf{p}_i$ and $\mathbf{p}_j$ are neighbours if the distance between them is smaller than a certain fixed value d.

[4]These tests have been executed in a PC with an Intel Core i7 CPU at 3.3 GHz, a NVIDIA GeForce GTX 470,

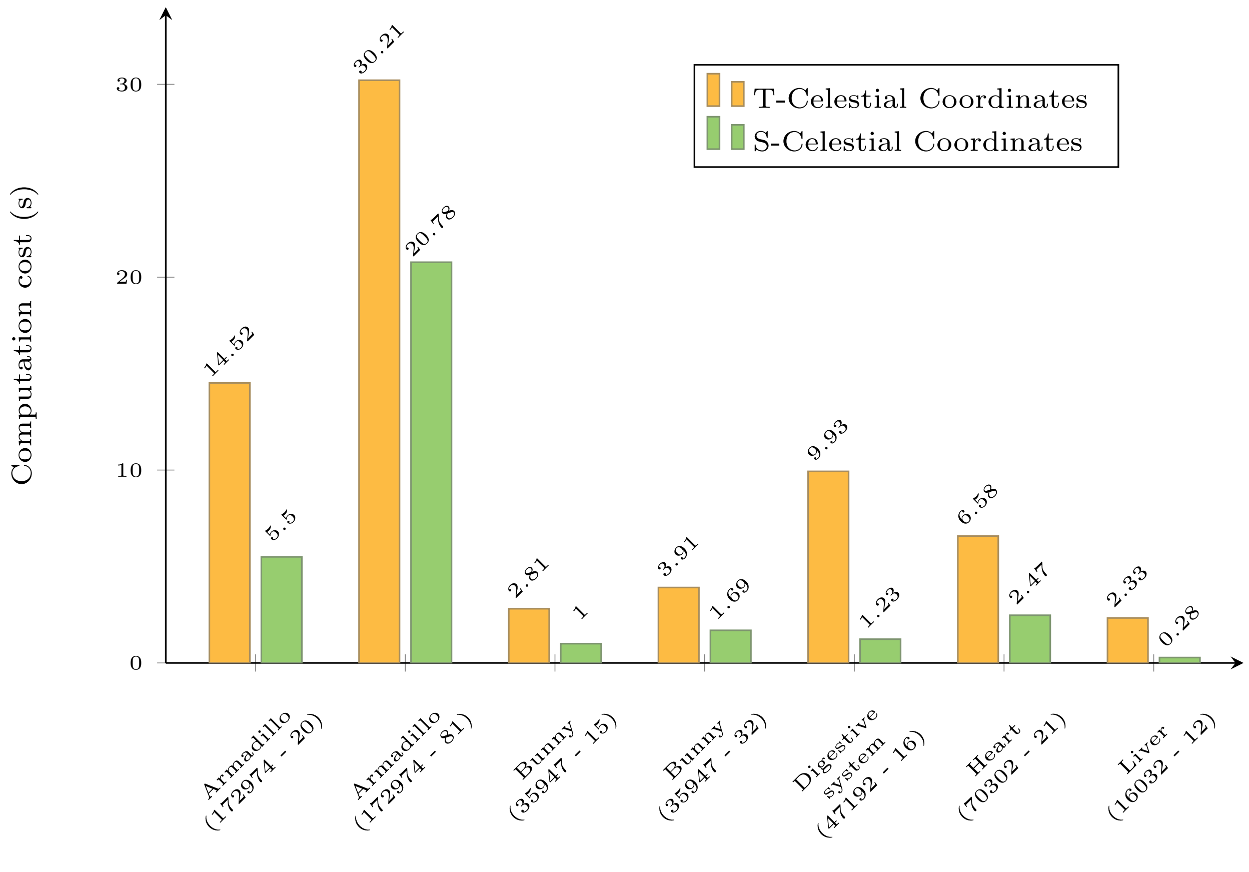


Figure 4.12: Comparison between the time, in seconds, used to compute the T-Celestial Coordinates and the S-Celestial Coordinates. This plot shows that the S-Celestial Coordinates are more efficient than the T-Celestial Coordinates. Additionally, below the name of each model are the number of points of that model and the number of control points of the configuration, in parenthesis.

Nonetheless, the S-Celestial Coordinates have the same global effect as the T-Celestial Coordinates, as shown in Figure 4.13. This drawback is particularly evident in the top image of Figure 4.13(b), where the control points around the ears of the *Bunny* also influence the chest and the back.

Finally, Figure 4.14 presents a comparison between those deformations that can be achieved by means of the T-Celestial Coordinates and those obtained with the help of the S-Celestial Coordinates. It shows four pairs of examples and each pair performs exactly the same deformation for both schemes of coordinates. We can observe that both approaches produce similar results. However, the deformations driven by the S-Celestial Coordinates are slightly smoother, as can be clearly noticed in the foot of the *Armadillo* model.

To conclude, the S-Celestial Coordinates scheme is efficient, rotational-invariant and also interpolates the control points, which make it appropriate and well-behaved for most applications. Table 4.1 in Section 4.6.1 compares the C-Celestial Coordinates, the T-Celestial Coordinates and the S-Celestial Coordinates according to these three properties.

6 GB of RAM and running Debian 8.0.

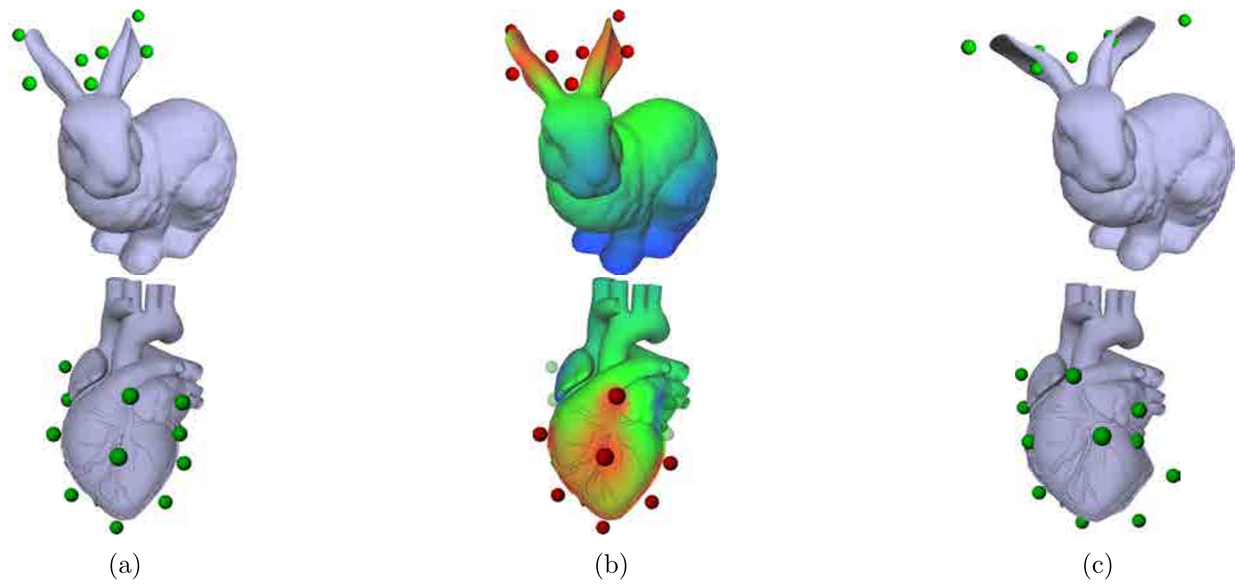


Figure 4.13: Example of two deformation processes driven by the S-Celestial Coordinates. (a) shows the initial configuration of the *Bunny* and the *Heart* models surrounded by deformation handles. Both models are enclosed in a cube-shaped convex hull, however, it can not be seen due to the zoom of the images. (b) presents the heatmap corresponding to the control points in red. The global effect of the coordinates can especially be seen in the *Bunny.* Observe that, although the red control points wrap the ears, the chest and the back are also affected by them. Finally, (c) shows the result of the deformation driven by the red handles in (b).

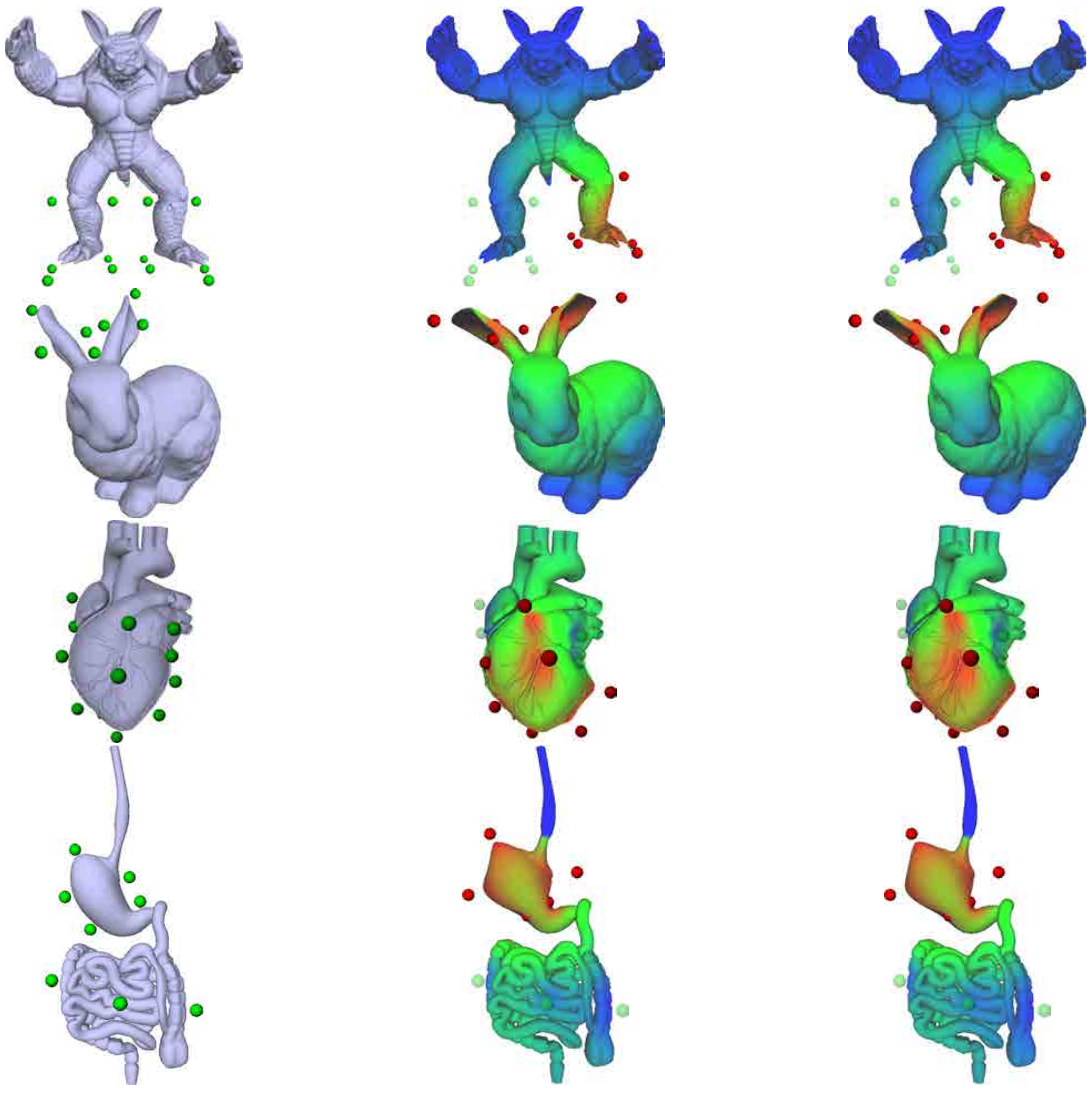
4.5 Localised Celestial Coordinates

The Celestial Coordinate schemes presented in this chapter suffer from the global effect of their coordinates. They are also unaware of the topology of the model enclosed in the convex hull of the control points. However, some applications may need more localised deformations than those achieved with the help of the C-Celestial Coordinates, the T-Celestial Coordinates and the S-Celestial Coordinates. Therefore, we sought a method to isolate the effect of the coordinates presented in the previous sections.

Our idea was to restrict the influence of the deformation handles by means of user-defined regions. These regions would provide user-intended shape-awareness and they could be adjacent or even overlap each other. Thus, we needed to guarantee that the coordinates remained continuous across their boundaries, so no artifacts appeared on the deformed 3D models.

In order to meet this challenge, we analysed a similar methodology to that presented by García et al. [GGPCP13]. *Cages were proposed as a multi-cage system. They use a blending function to increase the smoothness of the coordinates when crossing from one cage to an adjacent one. This blending takes into account not only the adjacent cages, but also their union.

Similarly, our approach blends local coordinates, those computed inside an independent region, with global coordinates, computed inside the union of independent regions. Notice that we replace the term *cage* by the idea of *region.* Hence, we first need to describe how to define a region.



(a) Initial configuration (b) TCC examples (c) SCC examples

Figure 4.14: Example of four deformation processes to compare the T-Celestial Coordinates with the S-Celestial Coordinates. (a) shows the initial configuration of the models and their corresponding control points. Then, (b) presents the deformations driven by the T-Celestial Coordinates. Finally, (c) displays the deformations caused by the S-Celestial Coordinates. The control points in each example have been located exactly in the same position in the two deformation schemes. Notice that, although the paired deformations are very similar to each other, the S-Celestial Coordinates produce slightly smoother deformations than the T-Celestial Coordinates approach. This situation can be clearly observed on the foot of the *Armadillo* and the ears of the *Bunny*.

Groups and Regions

As stated before, our method uses some user-defined groups of control points to help to localise the Celestial Coordinates. First, the user classifies the handles $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ into different groups

$G_i \subseteq \mathcal{V}$, in such a way that $\cup_i G_i = \mathcal{V}$. Then, the method automatically computes a region R_i for each group G_i as its convex hull, so $R_i = ch(G_i)$. However, these regions do not need to be a partition of the convex hull of the handles $ch(\mathcal{V})$, but their volumes can intersect. Finally, the Celestial Coordinates are defined in all the space comprised inside the convex hull of a given set of control points. Therefore, each group G_i of handles affects all the model points $\mathbf{p}$ inside its corresponding region R_i. Figure 4.15 shows two examples of control points distributed into groups and their corresponding regions.

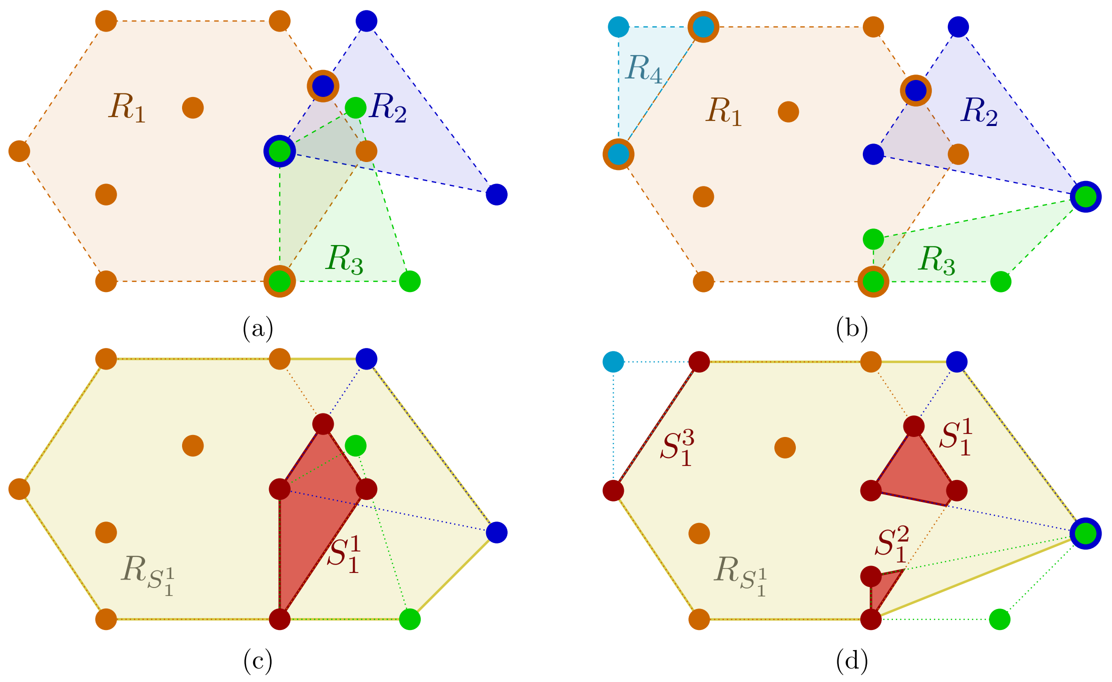


Figure 4.15: In (a) and (b) two different distribution of the control points in *Groups* and *Regions* are shown. On the other hand, (c) and (d) show the *Swapping Zones*, in red, and the *Super-Regions*, in yellow, associated to those distributions.

Once the concept *region* has been exposed, we have to ensure that crossing from one region to an adjacent one produces neither discontinuities on the coordinates nor visually unpleasant features on the deformed model. To this aim, we describe some more structures, which will help us in our purpose: the *adjacent regions*, the *swapping zones* and the *super-regions*.

Adjacent Regions. Two regions R_i and R_j are adjacent if, and only if, their intersection is not empty. That is

$$R_i \cap R_j \neq \emptyset \quad \text{for } i \neq j.$$

Swapping Zones. Given a region R_i, its swapping zones set $\mathcal{S}_i$ is composed of all the connected components that result from the intersections between R_i and the adjacent regions. That is

$$\mathcal{S}_i = \{\cup_j (R_i \cap R_j)\}.$$

As shown in Figures 4.15(a) and 4.15(c), the region R_1 has a single swapping zone $\mathcal{S}_1 = \{S_1^1\}$, which involves the regions R_1, R_2 and R_3. That is, $S_1^1 = (R_1 \cap R_2) \cup (R_1 \cap R_3)$. Additionally, Figures 4.15(b) and 4.15(d) show the region R_1 with three swapping zones $\mathcal{S}_1 = \{S_1^1, S_1^2, S_1^3\}$, where $S_1^1 = R_1 \cap R_2$, $S_1^2 = R_1 \cap R_3$ and $S_1^3 = R_1 \cap R_4$.

Notice that the intersections can result in swapping zones with positive volume, e.g. S_1^1 in Figure 4.15(c), or can make them degenerate into faces, e.g. S_1^3 in Figure 4.15(d).

Super-Regions. Any super-region $R_{S_i^k}$ is defined with respect to the region R_i and its swapping zone S_i^k. It is computed as the convex hull that results from the union of the groups G_j related to the regions R_j, which, in turn, are involved in the swapping zone S_i^k. Therefore, the set $\mathcal{R}_i$, that consists of all the super-regions corresponding to the region R_i, has the same cardinality as the set $\mathcal{S}_i$ of swapping zones. That is, there exists one super-region for every swapping zone.

Figures 4.15(a) and 4.15(c) show a single super-region $\mathcal{R}_1 = \{R_{S_1^1}\}$ for R_1. It is the convex hull of the groups G_1, G_2 and G_3, which correspond to the regions R_1, R_2 and R_3 involved in the swapping zone S_1^1. In contrast, Figures 4.15(b) and 4.15(d) show three super-regions $\mathcal{R}_1 = \{R_{S_1^1}, R_{S_1^2}, R_{S_1^3}\}$, where $R_{S_1^1} = ch(G_1 \cup G_2)$, $R_{S_1^2} = ch(G_1 \cup G_3)$ and $R_{S_1^3} = ch(G_1 \cup G_4)$.

Once the notions of group, region, swapping zone and super-region have been presented, we expose the procedure to ensure local, continuous and smooth Celestial Coordinates across these regions. We propose using a blending function in the vicinity of the swapping zones. Therefore, given two different regions R_i and R_j, the corresponding swapping zone $S_i^k = R_i \cap R_j$ and the related super-region $R_{S_i^k} = ch(G_i \cup G_j)$, the system fixes an offset distance h_i around the boundary of S_i^k. Then, for a particular model point $\mathbf{p}$ that belongs to the region R_i and also lies inside the volume in between the swapping zone S_i^k border and the perimeter defined by the distance h_i, we compute its coordinates as a blending between the coordinates with respect to the region R_i and the coordinates with respect to the super-region $R_{S_i^k}$. Accordingly, if we define $d = dist(\mathbf{p}, S_i^k)$ as the distance from the point $\mathbf{p}$ to the swapping zone S_i^k, the blending weights ρ_i are computed by means of a cubic Bézier curve, so

$$\begin{cases} \rho_i = 1 & \text{if } \mathbf{p} \text{ inside } S_i^k \\ \rho_i = \left(1 - \frac{d}{h_i}\right)^3 + 3\left(1 - \frac{d}{h_i}\right)^2 \left(\frac{d}{h_i}\right) & \text{else.} \end{cases}$$

Then, the blending function over the coordinates is the following

$$\boldsymbol{\beta} = \rho_i \boldsymbol{\beta}_{R_{S_i^k}} + (1 - \rho_i)\boldsymbol{\beta}_{R_i},$$

where $\boldsymbol{\beta}_{R_{S_i^k}}$ are the coordinates of $\mathbf{p}$ with respect to the super-region $R_{S_i^k}$ and $\boldsymbol{\beta}_{R_i}$ are the coordinates of $\mathbf{p}$ with respect to the region R_i.

In addition, the fixed distance h_i plays an important role in the behaviour of the blending function, representing the balance between the smoothness and the locality of the coordinates. Hence, higher values of h_i mean smoother and more global coordinates, since the super-region $R_{S_i^k}$ has a higher impact over the final set of coordinates. On the contrary, lower values of h_i result in more local coordinates. However, the changes on these coordinates when the point $\mathbf{p}$ approaches the swapping zone S_i^k are more abrupt, although continuous. Apart from this, we define an upper bound for the value h_i. For each swapping zone $S_i^k \in \mathcal{S}_i$, we first compute the minimum distance d_k between its boundary and the deformation handles that belong to the group G_i and are not part of S_i^k. Then, the upper bound of h_i is the minimum distance d_k found. That is

$$\begin{aligned} & d_k = \min dist(\mathbf{v}_j, S_i^k) && \forall \mathbf{v}_j : \mathbf{v}_j \in G_i \wedge \mathbf{v}_j \notin S_i^k \text{ and } \forall S_i^k \in \mathcal{S}_i \\ & h_i < \min\{d_k\}. \end{aligned}$$

Figure 4.16 illustrates the behaviour of the localised Celestial Coordinates through the different regions defined by the user.

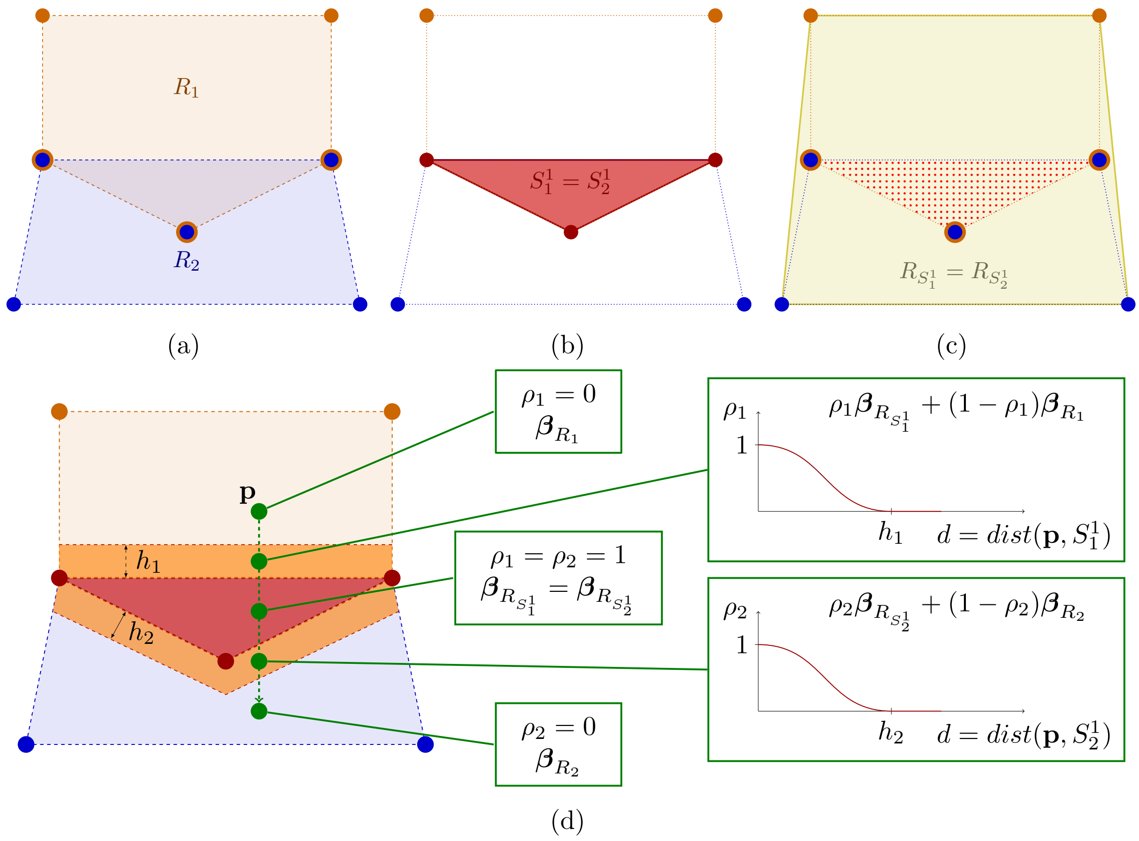

R_1
R_2
$S_1^1 = S_2^1$
$R_{S_1^1} = R_{S_2^1}$
(a)
(b)
(c)
$\mathbf{p}$
h_1
h_2
$\rho_1 = 0$
$\boldsymbol{\beta}_{R_1}$
$\rho_1 = \rho_2 = 1$
$\boldsymbol{\beta}_{R_{S_1^1}} = \boldsymbol{\beta}_{R_{S_2^1}}$
$\rho_2 = 0$
$\boldsymbol{\beta}_{R_2}$
ρ_1
$\rho_1\boldsymbol{\beta}_{R_{S_1^1}} + (1-\rho_1)\boldsymbol{\beta}_{R_1}$
1
h_1
$d = dist(\mathbf{p}, S_1^1)$
ρ_2
$\rho_2\boldsymbol{\beta}_{R_{S_2^1}} + (1-\rho_2)\boldsymbol{\beta}_{R_2}$
1
h_2
$d = dist(\mathbf{p}, S_2^1)$
(d)

Figure 4.16: In (a) the initial distribution of the control points in two *Groups* and their associated *Regions* are shown, R_1 on the upper part and R_2 on the lower part. In (b) and (c) the shared *Swapping Zone* and its *Super-Region* are highlighted. Finally, (d) represents a model point $\mathbf{p}$ that moves across the *Regions*. In each stage the blending factor ρ and the corresponding coordinates $\boldsymbol{\beta}$ are shown, so that ρ_1 is the blending factor with respect R_1 and ρ_2 is the blending factor corresponding to R_2. Moreover, $\boldsymbol{\beta}_{R_1}$ represents the Celestial Coordinates with respect to R_1, while $\boldsymbol{\beta}_{R_2}$ and $\boldsymbol{\beta}_{R_{S_1^1}} = \boldsymbol{\beta}_{R_{S_2^1}}$ are the Celestial Coordinates corresponding to R_2 and to the *Super-Region* $R_{S_1^1}$, which is the same as $R_{S_2^1}$.

4.5.1 Discussion

The localisation of the Celestial Coordinates allows us to focus the deformation process in a user-intended manner. The user can easily classify the control points into groups to get the best distribution to fit the desired deformation. Then, the system automatically computes the regions and their swapping-zones to ensure that the Celestial Coordinates remain continuous inside all the deformation domain, that is, inside the convex hull of the control points. Figure 4.17 shows how the influence area of a particular control point changes depending on whether the handles are classified into multiple groups or not.

Although we present the localisation strategy to work together with the C-Celestial Coordinates, the T-Celestial Coordinates and the S-Celestial Coordinates, it can also be applied together with the coordinate systems previously described in Chapter 3.

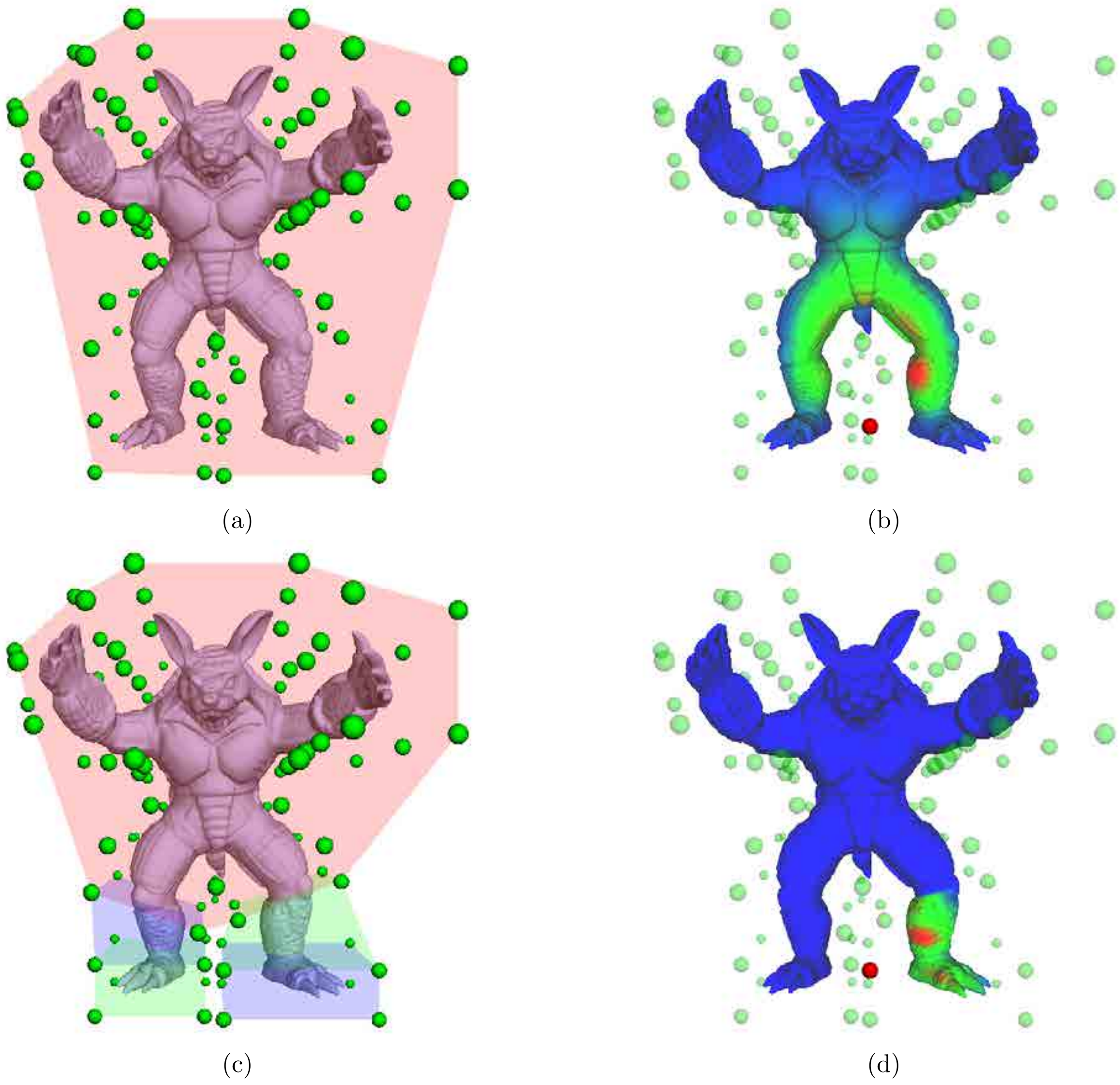


Figure 4.17: Two different distributions of the control points into *Groups* around the *Armadillo* and their corresponding heat maps. (a) and (c) show the corresponding *Regions*, each region in a different color. (b) and (d) show the heat map for the same control point in red. While the control point in (b) influences a large portion of the model, the same handle in (d) has a smaller and more localised impact area due to the effect of the multiple *Regions*.

4.6 Analysis and Comparison

This section is divided into two parts. First, Section 4.6.1 analyses and discusses the three Point-Based Celestial Coordinates presented in this chapter. Then, Section 4.6.2 compares our proposed Celestial Coordinate systems with some of the existing schemes.

4.6.1 Analysis of the Point-Based Celestial Coordinates

The three Point-Based Celestial Coordinate systems presented in the previous sections are well-defined, i.e. they fulfil all the properties of a Generalised Barycentric Coordinate system. Additionally, we want to emphasise some important features that they also show:

- **Positivity:** the Point-Based Celestial Coordinates and, particularly, the C-Celestial Coordinates, the T-Celestial Coordinates and the S-Celestial Coordinates are always positive inside the convex hull of the control points. This convex hull is the domain of the deformation.
- **Continuity and moothness:** the Point-Based Celestial Coordinates presented in this chapter are continuous and smooth.
- **Point-based user-intended awareness:** the Point-Based Celestial Coordinates allow the user to locate the control points wherever he or she considers more appropriate to achieve the desired deformation. The only requirement is that the model to be deformed must lie inside the convex hull of the deformation handles. Apart from that, the user can distribute the handles in the vicinity of those parts of the model on which he or she wants to focus.
- **Non-locality:** the C-Celestial Coordinates, the T-Celestial Coordinates and the S-Celestial Coordinates suffer from global impact. The modification of any control point leads to a deformation that affects all the points inside the convex hull of these handles. Obviously, the deformation has a bigger impact on the vicinity of the translated control point. However, it can affect distant regions and show non monotonic influence in some cases (see Figure 4.11). These kind of modifications do not take into account the topology of the wrapped model, so the deformation is not shape-aware.

In addition, the defined systems differ from each other in three main features, summarized in Table 4.1. On the one hand, the C-Celestial Coordinates have the main disadvantage of their inefficient computation (see Figure 4.18), whereas the T-Celestial Coordinates are not rotational-invariant. Therefore, the more appropriate and well-behaved Point-Based Celestial Coordinate system is the S-Celestial Coordinate scheme, which is efficient, rotational-invariant and also interpolates the control points.

	CCC	**TCC**	**SCC**
Rotational-invarian	✓	✗	✓
Interpolate control points	✓	✓	✓
Efficient	✗	✓	✓

Table 4.1: Comparison of the attributes that distinguish the three presented Point-Based Celestial Coordinates: the C-Celestial Coordinates (CCC), the T-Celestial Coordinates (TCC) and the S-Celestial Coordinates (SCC).

Figure 4.18 shows the computational cost to compute the C-Celestial Coordinates, the T-Celestial Coordinates and the S-Celestial Coordinates for the classic example of a line surrounded by fifteen control points. The model of the line consists of one thousand points and the experiments have been exercuted in a PC with an Intel Core i7 CPU at 3.3 GHz, a NVIDIA GeForce GTX 470, 6 GB of RAM and running Debian 8.0.

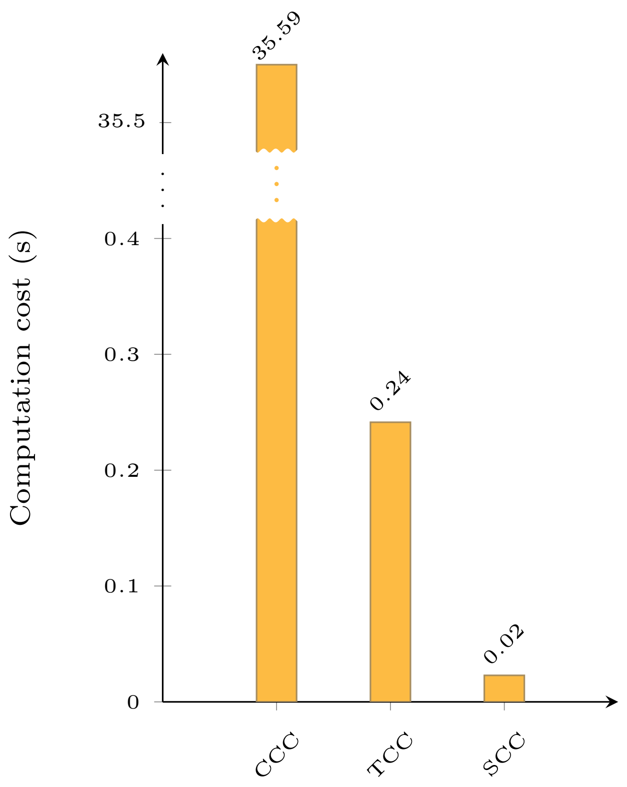


Figure 4.18: Comparison between the time, in seconds, used to compute the C-Celestial Coordinates (CCC), the T-Celestial Coordinates (TCC) and the S-Celestial Coordinates (SCC) for the configuration that consists in a line of one thousand points surrounded by fifteen control points, as shown in Figures 4.1, 4.4(b) and 4.11(b). This plot shows that the S-Celestial Coordinates are the most efficient of the Point-Based Celestial Coordinates presented in this chapter. Moreover, the cost of the C-Celestial Coordinates have been obtained for the example shown in Figure 4.1(a), which uses 10^5 Monte-Carlo samples.

Figure 4.19, as previously Figure 4.14, shows some examples of deformation processes that use the Point-Based Celestial Coordinates. The flexibility of the deformation handles combined with the efficiency of the Space Deformation paradigm make the Point-Based Celestial Coordinates very suitable to be applied in a medical context. In medical applications, doctors want interactive response to their interaction and also appreciate the freedom to locate the deformation handles wherever they consider more suitable, so they can experiment or simulate soft organs and tissue deformations. Accordingly, Figures 4.20, 4.21 and 4.22 show the effect of the Celestial Coordinates on some soft organs.

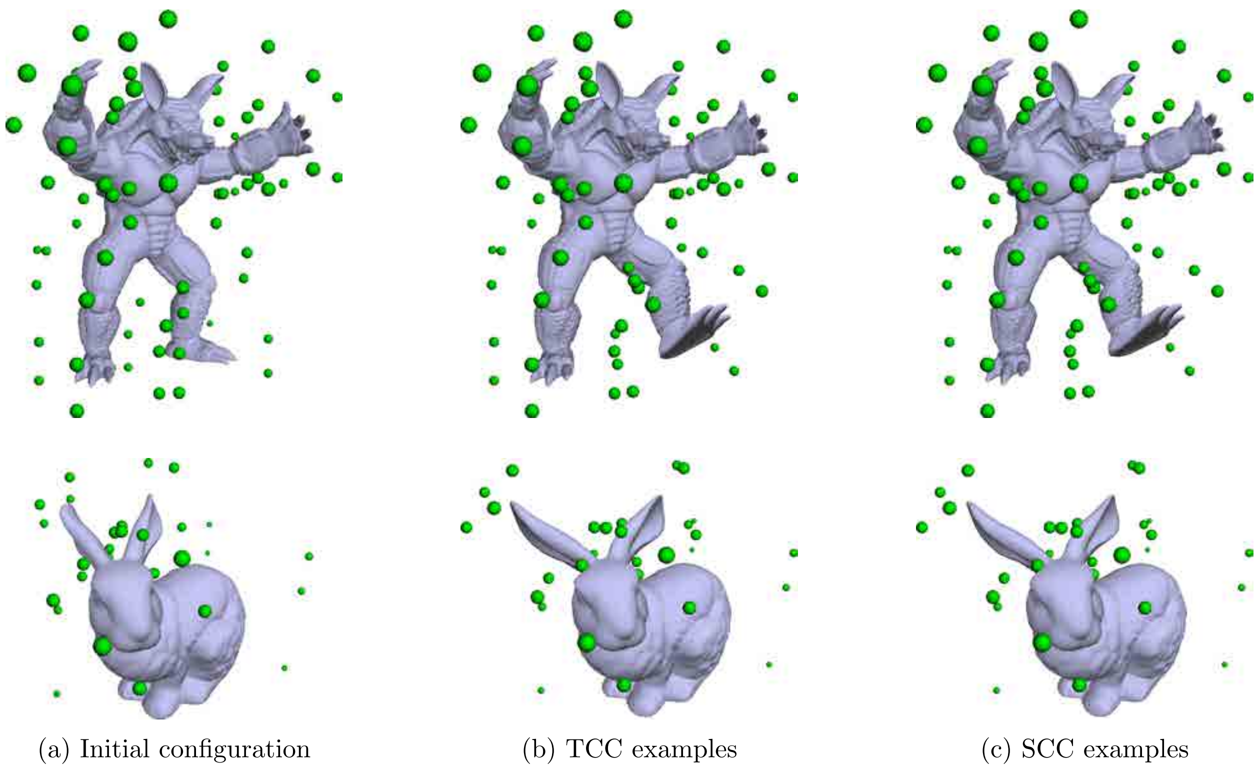

(a) Initial configuration (b) TCC examples (c) SCC examples

Figure 4.19: Examples of deformations with the T-Celestial Coordinates and the S-Celestial Coordinates. In (a) the initial situation and the control point distribution around the models are shown. The top images of (b) and (c) show the *Armadillo* model in a walking pose while the bottom images show the *Bunny* with folded ears.

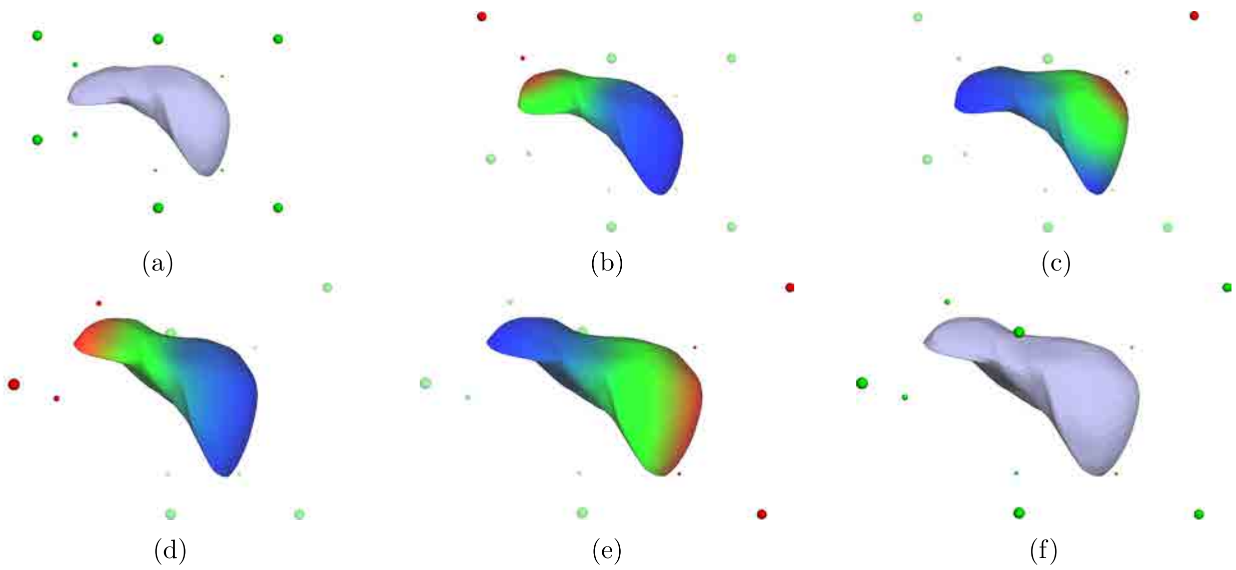

(a) (b) (c) (d) (e) (f)

Figure 4.20: Example of the deformation of the *Liver* model by means of the S-Celestial Coordinates. In (a), the initial situation with the control points around the model is shown. Images from (b) to (e) show a deformation sequence with the active handles in red and the corresponding heat maps. Finally, (f) shows the final result of the deformation.

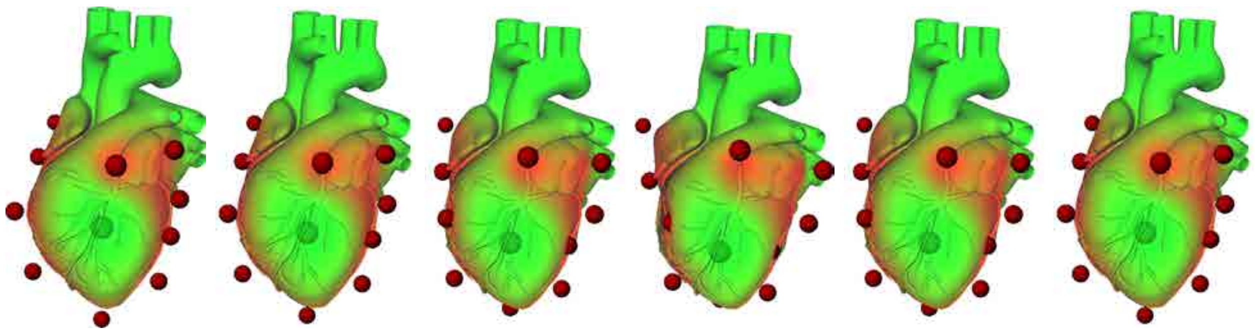


Figure 4.21: Example of a heartbeat deformation sequence by means of the S-Celestial Coordinates.

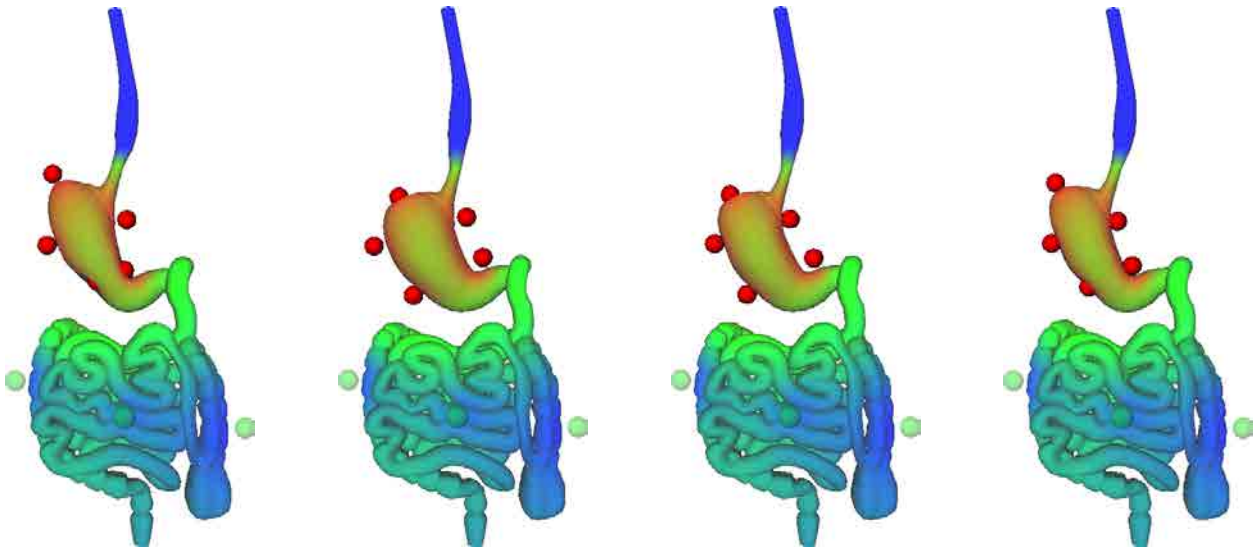


Figure 4.22: Example of a deformation sequence of the stomach movement by means of the S-Celestial Coordinates.

4.6.2 Comparison with some Existing Schemes

This section enumerates the similarities and the differences between the presented Point-Based Celestial Coordinates and some of the existing techniques. In particular, we compare the C-Celestial Coordinates (CCC), the T-Celestial Coordinates (TCC) and the S-Celestial Coordinates (SCC) with the Mean Value Coordinates [FKR05, JSW05] (MVC) and the Bounded Biharmonic Weights [JBPS11] (BBW).

All the aforementioned coordinate systems meet the basic properties of reproduction of the identity, reproduction of the unity, interpolation of the control points and positivity, although the MVC are only positive inside the kernel of the user-defined cage. They are also continuous and smooth inside the domain of the deformation.

Our Point-Based Celestial Coordinates are cage-free methodologies while the MVC are used in a cage-based paradigm. This allows these two systems of coordinates to have an easy formulation and their computation is efficient, except for the CCC, which have a high computational cost.

On the contrary, the BBW have the advantatge of combining multiple types of handles, so they can use skeletons, cages and isolated points. This feature makes this methodology more flexible. However, it increases both the complexity and the cost of the computation of the weights, as shown in Figure 4.23. Moreover, the BBW need to discretise the deformation domain together with the model and, therefore, the weights depend on how this discretisation is done.

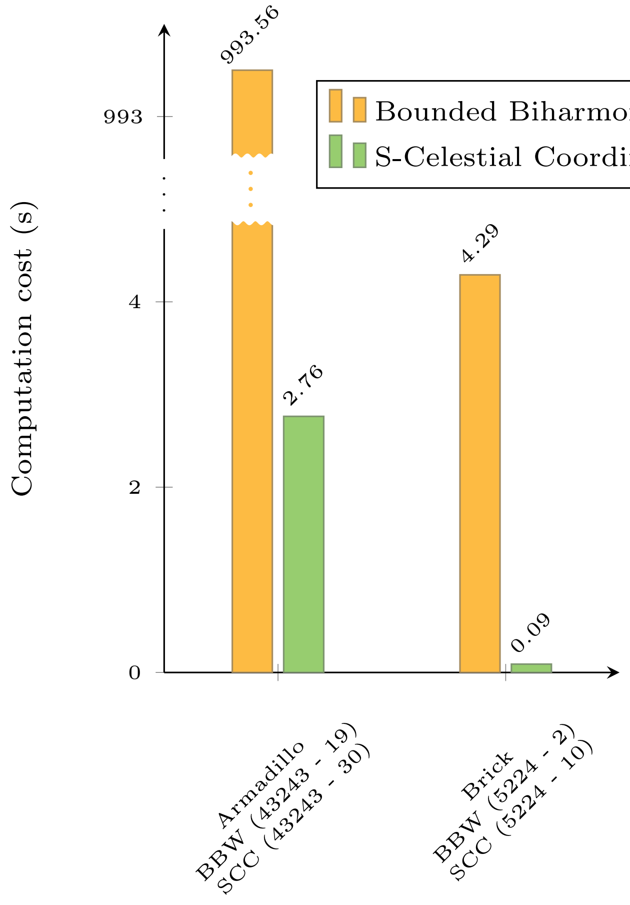


Figure 4.23: Comparison between the time, in seconds, used to compute the Bounded Biharmonic Weights and the S-Celestial Coordinates. This plot shows that the S-Celestial Coordinates are more efficient than the Bounded Biharmonic Weights. Additionally, below the name of each model are the number of points of that model and the number of control points of both configurations, in parenthesis. The examples are those provided by the *libigl* library [JP$^+$15] and the tests have been executed in a PC with an Intel Core i7 CPU at 3.3 GHz, a NVIDIA GeForce GTX 470, 6 GB of RAM and running Debian 8.0.

Cage-based methods suffer from non-locality of the deformation, a drawback that also concerns our coordinate systems. Nonetheless, this disadvantatge is minimized when we combine these methodologies with other techniques that localise the deformation, such as those presented in [GGPCP13] and in Section 4.5. Related to the non-locality feature, our Point-Based Celestial Coordinates can present non monotonic influence. In contrast, the BBW have been experimentally proven local andm monotonic.

Although the CCC, the TCC and the SCC allow the user to locate the control points wherever he or she considers more appropriate to achieve the desired deformation, they can distort the local details of the deformed model. A similar situation also occurs with the MVC. The BBW tend to preserve these details better due to the customised energy minimisation performed to compute their weights.

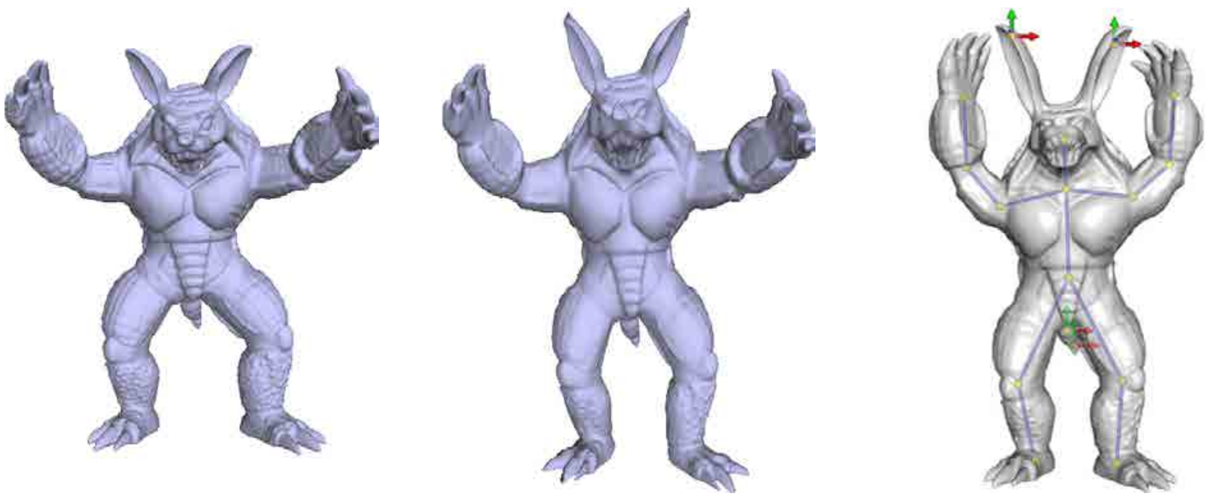
Table 4.2 summarises all the exposed features for the MVC, the BBW and the presented Point-Based Celestial Coordinates.

	MVC	**BBW**	**Point-Based Celestial Coordinates**		
			CCC	**TCC**	**SCC**
Well-defined	✓	✓	✓	✓	✓
Continuous and smooth	✓	✓	✓	✓	✓
Multiple handles	✗	✓	✗	✗	✗
Cage-free	✗	✓	✓	✓	✓
Efficient	✓	✗	✗	✓	✓
Simple	✓	✗	✓	✓	✓
Do not need discretisation	✓	✗	✓	✓	✓
Local	✓	✓	✓	✓	✓
Detail-preserving	✗	✓	✗	✗	✗

Table 4.2: Comparison of the attributes of the Mean Value Coordinates (MVC), the Bounded Biharmonic Coordinates (BBW), the C-Celestial Coordinates (CCC), the T-Celestial Coordinates (TCC) and the S-Celestial Coordinates (SCC). The green check marks denote that the methodology meets the corresponding feature while the red crosses imply the contrary. The orange check marks mean that the technique mostly fulfils the characteristic.

Figure 4.24 shows two deformations of the *Armadillo* model: the first one is driven by the S-Celestial Coordinates and the second one has been performed by means of the Bounded Biharmonic Weights. The BBW deformation uses fifteen control points that shape a skeleton and four isolated handles located in the tail and the ears of the model. Our deformation uses thirty control points. The first eight are used to create a large convex hull that encloses the *Armadillo*. Then, the nineteen control points used in the BBW example are also added to our deformation. Finally, we have added three additional control points at the nose and at the end of the feet of the model. Notice that the S-Celestial Coordinates can achieve a similar deformation to that shown in the Bounded Biharmonic Weights example, especially at the legs, the ears and the mouth of the model. However, the arms cannot be emulated. This situation is caused by the different kind of handles used in each deformation. The BBW deforms the arms by means of an skeleton while our method deforms them by the use of isolated control points inside a convex hull. This difference causes that our methods cannot completely mimic large bones tranformations. Instead, the presented systems of coordinates produce flexible and plausible deformations of soft models, that is, models that do not possess an inner skeleton, as shown in Figures 4.20, 4.21 and 4.22

To summarise the comparison, we can state that the presented Point-Based Celestial Coordinates show an important improvement over the classic cage-based systems such as the Mean Value Coordinates. They make the deformation process much more flexible due to their cage-free paradigm. Additionally, they are also more intuitive than the Mean Value Coordinates because their coordi-



(a) Initial *Armadillo* (b) SCC (c) BBW

Figure 4.24: Comparison between the deformation obtained by means of the Bounded Biharmonic Weights (BBW) in (c) and the S-Celestial Coordinates (SCC) in (b). Both images use the same control points, those shown in (b). However, we need to enclose the *Armadillo* into a large convex hull to correctly compute the S-Celestial Coordinates. Additionally, we have also added two additional control points at the end of the feet to achieve a similar deformation of the legs to that shown by the BBW and one more control point at the nose of the model to open his mouth in a similar way to the BBW example. Although the ears have been enlarged correctly, the arms do not pose like in the BBW case. The image in (c) has been obtained from [JBPS11].

nate functions are always positive, so the deformations do not present unexpected artifacts.

Although our Point-Based Celestial Coordinates can not combine multiple kinds of handles as the Bounded Biharmonic Weights do, they are much more simple and efficient owing to their specialisation in only one kind of deformation controls, the isolated control points. They can carry out similar deformations to those achieved by the Bounded Biharmonic Weights with just a few more handles. In addition, we have shown that they are a good tool to perform flexible, efficient, pleasant and plausible deformations of soft organs, tissues and medical models.

4.7 Summary

This chapter presents the Celestial Coordinates, a family of positive Generalised Barycentric Coordinate systems that compute the coordinates of a point **p** with respect to the control points $\mathbf{v}_i$ by means of the projection of these handles over the *celestial sphere* of **p**. We focus our attention specifically on the Point-Based Celestial Coordinates, a subfamily of Celestial Coordinates that aggregates those schemes that do not need any kind of connectivity between the deformation handles. This group includes all the approaches presented in Section 3.3, which, unfortunately, do not meet our expectations. However, it also collects the three new Celestial Coordinate schemes described in the present chapter. They are the C-Celestial Coordinates, the T-Celestial Coordinates and the S-Celestial Coordinates.

The three Point-Based Celestial Coordinate systems presented in the previous sections are well-defined, positive, continuous and smooth, as shown in Figures 4.1, 4.4 and 4.11. However, they are non-local and the modification of a handle has impact on its vicinity but also on further parts of the model. This situation can be appreciated in the heatmaps of the *Digestive system*, Figure 4.7(b), and the *Bunny*, Figure 4.13(b). In order to alleviate the global effect of the Point-Based Celestial Coordinates, we have also described a method to localise the deformation process. This method consists in classifying the control points into different Groups to offer user-intended shape-awareness. Only those parts of the model lying inside the convex hull, or the Region, of a particular Group will be deformed by the handles belonging to it. The method also ensures that the coordinates remain continuous across these Regions. Figure 4.17 shows a comparison between the heat map that corresponds to the particular control point in red when there exists a single Group or, on the contrary, multiple Groups.

The presented Point-Based Celestial Coordinates show an important improvement over the classic cage-based. They make the deformation process much more flexible due to their cage-free paradigm. Additionally, they are also intuitive because their coordinate functions are always positive, so the deformations do not present unexpected artifacts. Although they can not completely mimic the large bones transformations performed by the Bounded Biharmonic Weights, the Point-Based Celestial Coordinates are much more simple and efficient. Moreover, they are very suitable to be applied in a medical context, where doctors want interactive response to their interaction and also appreciate the freedom to locate the deformation handles wherever they consider more suitable, so they can experiment or simulate soft organs and tissue deformations. Accordingly, Figures 4.20, 4.21 and 4.22 show the effect of the Celestial Coordinates on some soft organs.

Chapter 5

Volume-Preserving Deformations

"To those who ask what the infinitely small quantity in mathematics is, we answer that it is actually zero. Hence, there are not so many mysteries hidden in this concept as they are usually believed to be."

Leonhard Euler

Chapter 5 Contents

Some applications require deforming 3D models under certain restrictions to preserve some of their geometric properties. In particular, those deformations that preserve the volume of the modified object are very important in *animation*, where artists want to create physically plausible motions and need to simulate muscles that keep their volume constant, or in *medicine*, where training applications involve the analysis of soft but incompressible organs and tissues.

As previously observed in Chapter 2, space deformation paradigm has received considerable attention in recent years. Particularly, cage-based methods are efficient and widely used to perform 3D model deformations. Hence, an important amount of Generalised Barycentric Coordinates (GBC) have appeared in order to manipulate the models enclosed in a coarse polyhedron. However, there are few GBC systems that are actually designed to produce volume-preserving deformations.

In consequence, we wanted to study how to keep the volume of an object undergoing a cage-based deformation constant. Therefore, we needed to analyse the behaviour of the volume of the model during its manipulation process and, moreover, mathematically relate it with the displacement of the deformation handles.

This chapter presents the mathematical generalisation of the volume preservation in the case of cage-based deformations. Furthermore, as we will see later, the described technique is not constrained to the existence of a cage, but it is also extensible to any deformation system that relies on a set of handles surrounding the model. Section 5.1 develops the mathematical concepts of volume preservation and Section 5.2 provides a tool to compute the stress that the deformation induces over the model. Finally, Section 5.3 summarises the contents of the chapter.

The research exposed has been compiled in [CVB10] and successfully used in [CHHH15].

5.1 Gradient-Based Volume-Preserving Deformations

Given a particular model, whose surface is represented as a triangular mesh, enclosed in a cage, we are interested in mathematically finding the connection between the displacement of the cage vertices, the handles $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, and the deviation of its volume. Therefore, every time the user picks a control point, we want to establish a path or a surface such that the movement of the handle over it does not cause any perturbation of the volume of the deformed model.

The gradient of a function points in the direction of its greatest increment. Accordingly, the gradient of the volume expresses its greatest perturbation direction. Thus, a handle moving along the gradient will dramatically change the volume of the object while deforming it. Our idea, then, is to restrict the movement of the control points to those surfaces perpendicular to this gradient.

The following sections give mathematical expressions to compute the volume of a given object and, then, derive the formula of its gradient. We also study the behaviour of the gradient during the deformation process.

5.1.1 Mesh Volume and Gauss Theorem

The Gauss Theorem [Apo69], or the Divergence Theorem, states that the volume integral of the divergence of a vector field inside a closed region is equivalent to the surface integral of the same vector field over the boundary of this region. That is

$$\int\int\int_V \nabla\mathbf{F}\, dV = \int\int_S \mathbf{F} \cdot \vec{\mathbf{n}}\, dS, \tag{5.1}$$

where $\mathbf{F}$ is the vector field computed on the points of the surface S and $\vec{\mathbf{n}}$ are the outward unit normals at these points.

In the case we are interested in, we chose the vector field $\mathbf{F}(\mathbf{p}) = \mathbf{p}$, for any 3D point $\mathbf{p} = (x, y, z)$. Then, as it can be noticed in the mathematical development below, when $\mathbf{F}$ is substituted in Equation (5.1), it results in computing the volume of the integration region multiplied by three

$$\begin{aligned} \int\int\int_V \nabla\mathbf{F}\, dV &= \int\int\int_V \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \cdot (x, y, z)\ dV \\ &= 3\int\int\int_V dV. \end{aligned}$$

Therefore, the volume V of a given model can be computed as

$$V = \frac{1}{3}\int\int\int_V \nabla\mathbf{F}\, dV = \frac{1}{3}\int\int_S \mathbf{F} \cdot \vec{\mathbf{n}}\, dS. \tag{5.2}$$

However, our models are not defined by an analytical surface, but their boundary is discretised by means of a triangular mesh. For this reason, we needed to numerically integrate $\mathbf{F}$ over the mesh. We propose using Gauss Numerical Integration, or the Gaussian Quadrature, [Abr74] to face this process. This method offers numerical stability and absence of truncation errors for polynomial functions and consists in evaluating the function in some precomputed integration points. Therefore, the surface integration in Equation (5.2) is discretised as

$$V = \frac{1}{3}\int\int_S \mathbf{F}\cdot\vec{\mathbf{n}}\, dS = \frac{1}{3}\sum_{j=1}^{m}\left[a_j\vec{\mathbf{n}}_j\cdot\left(\sum_{k=1}^{l}\phi_k p_k\right)\right], \tag{5.3}$$

where m is the number of triangles of the mesh, a_j is the area of the triangle j, $\vec{\mathbf{n}}_j$ is the outward unit normal of the same triangle, l are the number of gaussian integration points, p_k is the k-th integration point over the triangle and ϕ_k is its corresponding gaussian weight. The points p_k are defined as

$$p_k = \eta_{k_1}\mathbf{p}_{j_1} + \eta_{k_2}\mathbf{p}_{j_2} + \eta_{k_3}\mathbf{p}_{j_3}, \tag{5.4}$$

where $\mathbf{p}_{j_1}$, $\mathbf{p}_{j_2}$ and $\mathbf{p}_{j_3}$ are the vertices of the triangle j and $(\eta_{k_1}, \eta_{k_2}, \eta_{k_3})$ are the gaussian barycentric coordinates of the integration points, defined in [Cow73] and [Dun85]. In addition, it is a well known fact that the magnitude of the outward normal $\vec{\mathbf{N}}$ of a triangle is equivalent to its area multiplied by two, that is, $\vec{\mathbf{N}} = 2a$. Therefore, Equation (5.3) can be simplified as

$$V = \frac{1}{6}\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_j\cdot\left(\sum_{k=1}^{l}\phi_k p_k\right)\right].$$

Finally, the vertices $\mathbf{p}$ of the triangular mesh are related to the vertices $\mathbf{v}_i$ of the cage through a Generalised Barycentric Coordinate system

$$\mathbf{p} = \sum_{i=1}^{n}\mathbf{v}_i\alpha_i(\mathbf{p}),$$

where n are the number of handles. Hence, the final formula for the volume is shown below

$$\begin{aligned} V &= \frac{1}{6}\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_j\cdot\left(\sum_{k=1}^{l}\phi_k p_k\right)\right] \\ &= \frac{1}{6}\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_j\cdot\left(\sum_{k=1}^{l}\phi_k(\eta_1\mathbf{p}_{j_1} + \eta_2\mathbf{p}_{j_2} + \eta_3\mathbf{p}_{j_3})\right)\right] \\ &= \frac{1}{6}\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_j\cdot\left(\sum_{k=1}^{l}\phi_k\left[\eta_1\sum_{i=1}^{n}\mathbf{v}_i\alpha_i(\mathbf{p}_{j_1}) + \eta_2\sum_{i=1}^{n}\mathbf{v}_i\alpha_i(\mathbf{p}_{j_2}) + \eta_3\sum_{i=1}^{n}\mathbf{v}_i\alpha_i(\mathbf{p}_{j_3})\right]\right)\right]. \end{aligned} \tag{5.5}$$

5.1.2 Mathematical Expression for the Gradient of the Volume

In the section above, we have brought out the analytical expression to compute the volume of a 3D model, whose surface is represented as a triangular mesh. Now, we want to study how this volume oscillates when a particular handle $\mathbf{v}_i$ of the cage is displaced a vector $\vec{\boldsymbol{\delta}}_i$. Thus, the increase, or decrease, of the volume is defined as

$$\Delta V = \vec{\boldsymbol{\delta}}_i \cdot (\nabla V)_{\mathbf{v}_i},$$

where $(\nabla V)_{\mathbf{v}_i}$ denotes the gradient of the volume with respect to $\mathbf{v}_i$ and V is defined by the Equation (5.5). Accordingly, the expression of the gradient of the volume is

$$(\nabla V)_{\mathbf{v}_i} = \left(\frac{\partial V}{\partial v_{i_x}}, \frac{\partial V}{\partial v_{i_y}}, \frac{\partial V}{\partial v_{i_z}} \right),$$

hence,

$$(\nabla V)_{\mathbf{v}_i} = \frac{1}{6} \left(\sum_{j=1}^{m} \left[(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_i} \cdot \left(\sum_{k=1}^{l} \phi_k p_k \right) \right] + \sum_{j=1}^{m} \left[\vec{\mathbf{N}}_j \cdot \left(\sum_{k=1}^{l} \phi_k (\nabla p_k)_{\mathbf{v}_i} \right) \right] \right), \tag{5.6}$$

where $(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_i} = (\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{i_x}}, \frac{\partial \vec{\mathbf{N}}_j}{\partial v_{i_y}}, \frac{\partial \vec{\mathbf{N}}_j}{\partial v_{i_z}})$ and $(\nabla p_k)_{\mathbf{v}_i} = (\frac{\partial p_k}{\partial v_{i_x}}, \frac{\partial p_k}{\partial v_{i_y}}, \frac{\partial p_k}{\partial v_{i_z}})$. For a complete mathematical development of the gradient, please refer to Annex B.

5.1.3 Behaviour of the Gradient of the Volume

Once the formula for the gradient of the volume has been exposed, we study how this gradient behaves when a control point is picked and dragged, that is, during the deformation process. To meet this goal, we, once again, assume that the picked handle is $\mathbf{v}_i$ and that it is translated a vector $\vec{\boldsymbol{\delta}}_i$, without loss of generality. In order to analyse the impact of the displacement of $\mathbf{v}_i$ by $\vec{\boldsymbol{\delta}}_i$ on the gradient, we split its definition into two components

$$(\nabla V)_{\mathbf{v}_i} = (\nabla V)_{\mathbf{v}_{i,i}} + (\nabla V)_{\mathbf{v}_{i,\bar{i}}},$$

where $(\nabla V)_{\mathbf{v}_{i,i}}$ clusters together the terms that contain $\mathbf{v}_i$ and $(\nabla V)_{\mathbf{v}_{i,\bar{i}}}$ the remaining ones. Then, Equation (5.6) can also be expressed as

$$\begin{aligned}(\nabla V)_{\mathbf{v}_i} = \frac{1}{6} \Bigg[\Bigg(& \sum_{j=1}^{m} \left[(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{i,i}} \cdot \left(\sum_{k=1}^{l} \phi_k p_{k,i} \right) \right] + \sum_{j=1}^{m} \left[(\nabla \vec{\mathbf{N}}_j))_{\mathbf{v}_{i,i}} \cdot \left(\sum_{k=1}^{l} \phi_k p_{k,\bar{i}} \right) \right] + \\ & + \sum_{j=1}^{m} \left[(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{i,\bar{i}}} \cdot \left(\sum_{k=1}^{l} \phi_k p_{k,i} \right) \right] + \sum_{j=1}^{m} \left[(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{i,\bar{i}}} \cdot \left(\sum_{k=1}^{l} \phi_k p_{k,\bar{i}} \right) \right] \Bigg) + \\ & + \Bigg(\sum_{j=1}^{m} \left[\vec{\mathbf{N}}_{j,i} \cdot \left(\sum_{k=1}^{l} \phi_k (\nabla p_k)_{\mathbf{v}_{i,i}} \right) \right] + \sum_{j=1}^{m} \left[\vec{\mathbf{N}}_{j,i} \cdot \left(\sum_{k=1}^{l} \phi_k (\nabla p_k)_{\mathbf{v}_{i,\bar{i}}} \right) \right] + \\ & + \sum_{j=1}^{m} \left[\vec{\mathbf{N}}_{j,\bar{i}} \cdot \left(\sum_{k=1}^{l} \phi_k (\nabla p_k)_{\mathbf{v}_{i,i}} \right) \right] + \sum_{j=1}^{m} \left[\vec{\mathbf{N}}_{j,\bar{i}} \cdot \left(\sum_{k=1}^{l} \phi_k (\nabla p_k)_{\mathbf{v}_{i,\bar{i}}} \right) \right] \Bigg) \Bigg].\end{aligned}$$

The two components are

$$(\nabla V)_{\mathbf{v}_{i,i}} = \frac{1}{6} \left(\sum_{j=1}^{m} \left[(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{i,i}} \cdot \left(\sum_{k=1}^{l} \phi_k p_k \right) \right] + \sum_{j=1}^{m} \left[(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{i,\bar{i}}} \cdot \left(\sum_{k=1}^{l} \phi_k p_{k,i} \right) \right] + \sum_{j=1}^{m} \left[\vec{\mathbf{N}}_{j,i} \cdot \left(\sum_{k=1}^{l} \phi_k (\nabla p_k)_{\mathbf{v}_i} \right) \right] \right)$$

$$(\nabla V)_{\mathbf{v}_{i,\bar{i}}} = \frac{1}{6} \left(\sum_{j=1}^{m} \left[(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{i,\bar{i}}} \cdot \left(\sum_{k=1}^{l} \phi_k p_{k,\bar{i}} \right) \right] + \sum_{j=1}^{m} \left[\vec{\mathbf{N}}_{j,\bar{i}} \cdot \left(\sum_{k=1}^{l} \phi_k (\nabla p_k)_{\mathbf{v}_{i,\bar{i}}} \right) \right] \right).$$

Then, if we expand the expression for $(\nabla V)_{\mathbf{v}_{i,i}}$, as done in Annex B, Section B.2, we find that

$$(\nabla V)_{\mathbf{v}_{i,i}} = \frac{1}{6}\left(\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{i,i}} \cdot \left(\sum_{k=1}^{l}\phi_k p_k\right)\right] + \sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{i,\bar{i}}} \cdot \left(\sum_{k=1}^{l}\phi_k p_{k,i}\right)\right] + \right.$$
$$\left. + \sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{j,i} \cdot \left(\sum_{k=1}^{l}\phi_k \nabla p_k\right)\right]\right) = \mathbf{0}. \tag{5.7}$$

Two important facts arise from Equation (5.7). The first one states that the gradient of the volume does not depend on the modified control point $\mathbf{v}_i$, but it only depends on the handles that stay in their positions while the vertex $\mathbf{v}_i$ is dragged. Hence, $(\nabla V)_{\mathbf{v}_i} = (\nabla V)_{\mathbf{v}_{i,\bar{i}}}$. According to this, the second one asserts that this gradient is constant regardless of the translation $\vec{\boldsymbol{\delta}}_i$ applied to $\mathbf{v}_i$

$$(\nabla V)_{\mathbf{v}'_i} = (\nabla V)_{\mathbf{v}_i},$$

where we used the notation $(\nabla V)_{\mathbf{v}'_i}$ to refer to the gradient of the volume with respect to the displaced control point $\mathbf{v}'_i = (\mathbf{v}_i + \vec{\boldsymbol{\delta}}_i)$.

Our aim is to move the control point $\mathbf{v}_i$ in such a way that the volume of the deformed model remains constant. That is

$$\Delta V = \vec{\boldsymbol{\delta}}_i \cdot (\nabla V)_{\mathbf{v}_i} = 0. \tag{5.8}$$

Ergo, only those displacement vectors $\vec{\boldsymbol{\delta}}_i$ orthogonal to $(\nabla V)_{\mathbf{v}_i}$ will satisfy Equation (5.8). Since the gradient $(\nabla V)_{\mathbf{v}_i}$ is constant given a particular configuration of the cage, the surface over which the control point $\mathbf{v}_i$ can be dragged without changing the volume of the model is a plane that contains it and whose normal is the gradient $(\nabla V)_{\mathbf{v}_i}$ itself. Figure 5.1 shows an example of the control plane that ensures the volume preservation when the handle $\mathbf{v}_i$ is moved within it.

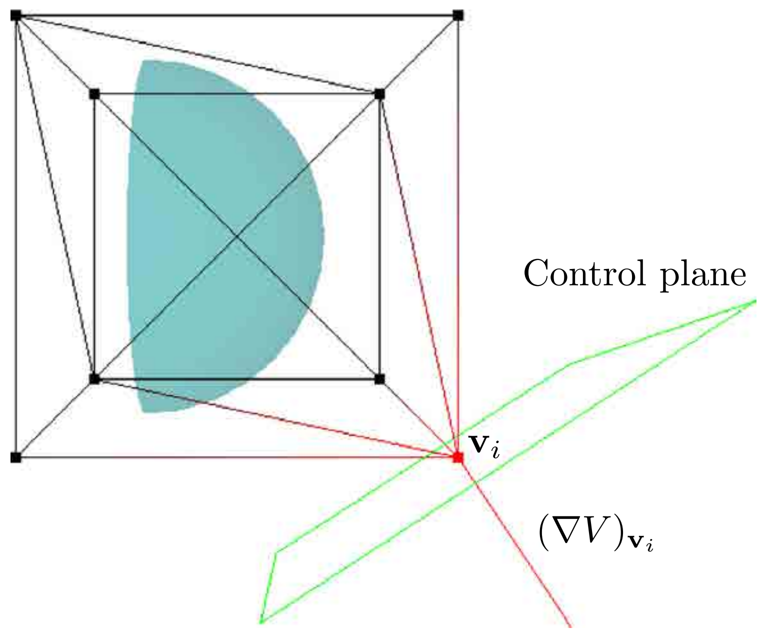


Figure 5.1: Control plane passing through the handle $\mathbf{v}_i$ and with normal $(\nabla V)_{\mathbf{v}_i}$. Any movement of the control point $\mathbf{v}_i$ over this plane preserves the volume of the deformed model.

5.1.4 Results and Discussion

We have presented the mathematical reasoning behind our proposal to produce volume-preserving cage-based deformations. Next, we give the algorithm to put the control of the volume into action and specify some details of its implementation. Finally, we discuss the achieved results.

According to Section 2.3, the cage-based deformation procedure is divided up into two stages: the *deformation set-up* and the *deformation process*. Hence, in order to extend the basic deformation with the volume-preserving capability, we needed to slightly modify its classic pipeline. The new routine is the following:

- *Deformation set-up:* this step remains exactly the same as in the classic pipeline. It consists in computing and storing the Generalised Barycentric Coordinates (GBC).
- *Deformation process:* every time that the user picks a handle $\mathbf{v}_i$, the mesh inside the cage is deformed in a two-pass algorithm. First, we compute the gradient of the volume $(\nabla V)_{\mathbf{v}_i}$ with respect to $\mathbf{v}_i$. Then, the displacements of the control point $\mathbf{v}_i$ are restricted to the control plane whose normal is $(\nabla V)_{\mathbf{v}_i}$. Thus, when the user drags the handle on it, the mesh is modified according to the new position of $\mathbf{v}_i$ and its corresponding GBC α_i, while its interior volume remains constant.

In order to test the proposed volume-preserving algorithm, we developed a deformation framework that traces the described procedure. Our implementation uses four gaussian integration points to compute the gradient of the volume (refer to Section 5.1), which, according to [Dun85], are obtained with the help of the weights and the barycentric coordinates shown in Table 5.1. Then, the integration point p_k is computed as in Equation (5.4).

Gaussian Weight	**Gaussian Barycentric Coordinates**		
ϕ_k	η_{k_1}	η_{k_2}	η_{k_3}
-0.5625	$0.\dot{3}$	$0.\dot{3}$	$0.\dot{3}$
	0.6	0.2	0.2
$0.5208\dot{3}$	0.2	0.6	0.2
	0.2	0.2	0.6

Table 5.1: The weights and barycentric coordinates to compute a Gaussian Quadrature over a triangle by means of four integration points. These values are defined in [Dun85].

Although our framework uses the Mean Value Coordinates (MVC) [FKR05], extensively detailed in Section 3.1, to illustrate the deformation examples, our volume-preserving cage-based deformation proposal works with all the GBC systems presented in Section 2.3. Moreover, it can also be applied to the cage-free schemes presented in Chapters 3 and 4. The only exception are the Green Coordinates (GC) [LLCO08, LL10]. As exposed previously, this GBC system does not only depend on the position of the handles, but also on the normals of the faces of the cage, so it defines the point $\mathbf{p}$ as the linear combination

$$\mathbf{p} = \sum_i \mathbf{v}_i \alpha_i(\mathbf{p}) + \sum_j \vec{\mathbf{n}}_j s_j \beta_j(\mathbf{p}).$$

For this reason, we should adapt the formulas of the volume and its gradient and derive this linear combination if we were interested in performing volume-preserving cage-based deformations with the GC. However, the mathematical reasoning behind our strategy remains the same, so we can affirm that our volume-preserving scheme also works together with the GC.

Table 5.2 shows the time cost of the computation of the volume and its gradient. It demonstrates that the gradient can be obtained interactively, thus, volume-preserving deformations can be performed in real-time. Notice also that the gradient is completely local to each point $\mathbf{p}$ of the model. Therefore, its computation is extremely parallelisable. Particularly, the times presented in Table 5.2 correspond to a GPU implementation executed in a PC with an Intel Core i7 CPU at 3.3 GHz, a NVIDIA GeForce GTX 470, 6 GB of RAM and running Debian 8.0. Additionally, the deformation framework has been implemented in C++.

	Vertices	**Triangles**	**Volume cost**	**Gradient cost**
Teapot	530	1024	10.744	0.193
Monkey	2868	968	10.887	1.854
Bunny	35947	69451	14.826	3.470
Digestive system	47192	89164	14.259	6.929
Heart	70302	133169	17.375	7.990
Armadillo	172974	345944	29.174	8.823

Table 5.2: Computation time of the volume and its gradient, in ms.

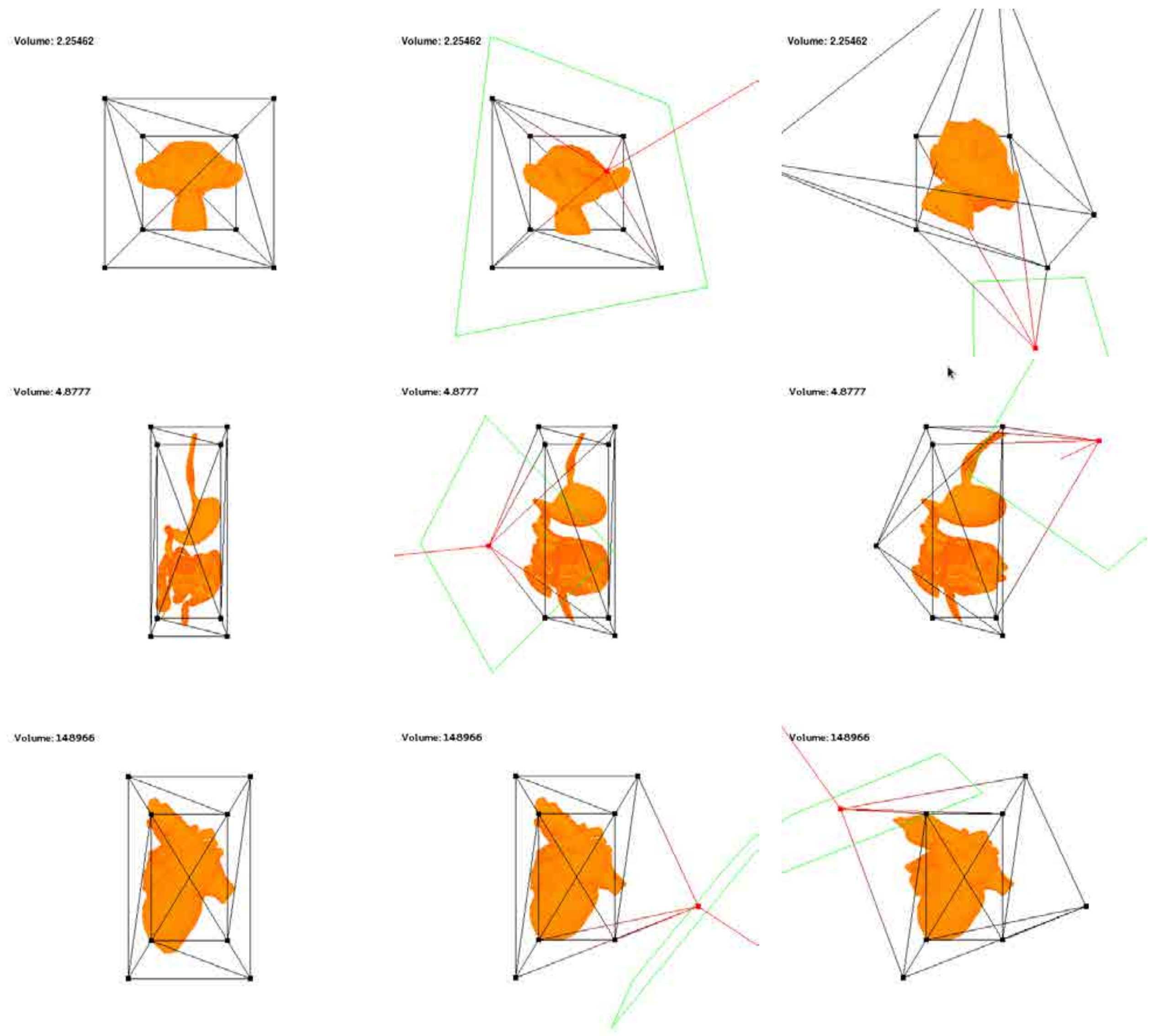
Figure 5.2 shows extreme deformations of different models. Notice how their interior volumes remain constant despite the strong modification of their surfaces.

5.1.5 Comparison with some Existing Schemes

This section presents a brief comparison between the proposed volume-preserving deformation method and the literature exposed in Section 2.4.

Most of the works devoted to study the volume-preserving deformations belong to the surface deformation paradigm [ZHS+05, vFTS06, BK03, LCOGL07] or to the skeleton-based deformation paradigm [vFTS08, AS07, RHC09] and they can not be directly applied to the space deformation methods. Instead of this, our technique has been specifically designed to achieve volume-preserving space deformations. Additionally, the techniques presented in [ZHS+05] and [BK03] only perform a local preservation of the volume to avoid unnatural changes in this property while our approach exactly preserves the volume of the deformed model. Furthermore, [LCOGL07] and [vFTS08] achieve the volume-preserving feature by means of a correction of the initial deformation defined by the user and our method directly constrains the deformation to meet the required volume preservation.

Research in [AB97], [HML99], [RSB95] and [WM14] preserve the volume in the Free-Form Deformation (FFD) paradigm. Specifically, [AB97] and [HML99] achieve this goal by means of an optimization problem. Although this works are space deformation techniques, they rely on the underlying lattice of the FFD method, which does not match the cage-based control structure as our approach does.



(a) (b) (c)

Figure 5.2: Examples of extreme volume-preserving deformations. (a) shows the initial state of the models, which includes the initial configuration of the cage and the initial value of the volume. (b) and (c) show two different snapshots of the deformations.

Finally, the methodology presented by Huang et al. [HSL$^+$06] preserves the volume of a cage-based deformation in a two-step algorithm. First, it modifies the cage so the volume of it is preserved by means of an energy minimisation. Then, the modification of the control structure is transferred to the model by means of a mean value interpolation. Instead, our approach preserves the volume directly by means of the gradient of this attribute. The computation of this gradient is very efficient and, in most cases, it can be obtained through a closed formula. Apart from this, the proposed method is more flexible and can be applied together with any kind of Generalised Barycentric Coordinate system. Therefore, it is not solely attached to the cage-based deformation paradigm but it also works with cage-free methods and, particularly, it can be adapted to the Point-Based Celestial Coordinates presented in Chapter 4.

5.2 Local Deformation Stress

It is possible that any deformation process and, particularly, those that try to preserve the volume of the modified object, causes some distortion and detail loss over the modified model, as we can see in Figure 5.2. However, this problem is not induced by the volume control mechanism, but it is intrinsic to the nature of the Generalised Barycentric Coordinate (GBC) system used in the deformation.

In this context, we studied the distortion that the deformation produces around a point $\mathbf{p}_i$ located inside the volume of the model. We defined an arbitrarily small sphere centered on $\mathbf{p}_i$ and computed a vector $\vec{\mathbf{r}} = (\mathbf{p}_s - \mathbf{p}_i)$ for each point $\mathbf{p}_s$ on its surface, as shown in Figure 5.3.

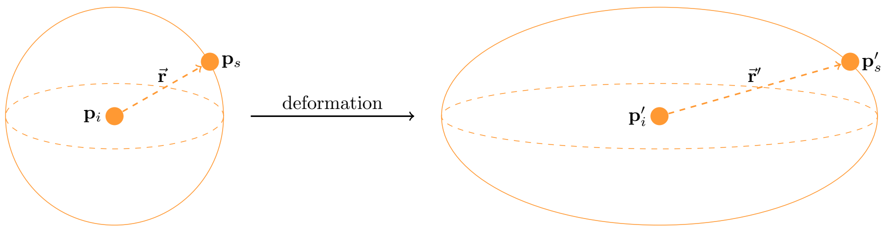


Figure 5.3: Example of the distortion that the deformation process can cause around an interior point $\mathbf{p}_i$ of the modified model.

Any cage-based deformation process modifies the points $\mathbf{p}$ of the model with the well-known linear combination

$$\mathbf{p} = \sum_i \mathbf{v}_i \alpha_i(\mathbf{p}).$$

Then, the vector $\vec{\mathbf{r}}$ is transformed according to a matrix M

$$\vec{\mathbf{r}}' = M \cdot \vec{\mathbf{r}},$$

whose columns correspond to the partial, or directional, derivatives of the GBC functions with respect to $\mathbf{p}$. That is

$$M = \sum_i \left[\frac{\partial \alpha_i(\mathbf{p})}{\partial p_x}, \frac{\partial \alpha_i(\mathbf{p})}{\partial p_y}, \frac{\partial \alpha_i(\mathbf{p})}{\partial p_z} \right] \cdot \mathbf{v}'_i.$$

The Singular-Value Decomposition of M returns three matrices $M = USV^\top$. U and V are two orthogonal matrices that produce rigid transformations, that is, they cause the same rotation on any vector $\vec{\mathbf{r}}$ and, therefore, do not introduce any local distortion around $\mathbf{p}_i$. In contrast, S is a diagonal matrix that consists of the singular values of M, $S = diag(s_1, s_2, s_3)$. It induces an anisotropic scale around $\mathbf{p}_i$ and causes a shear effect. Hence, we define the stretch, or rotational stress, σ at $\mathbf{p}_i$ as the maximum rotation introduced by S

$$\sigma = \max_{\vec{\mathbf{r}}} \left[\arccos \left(\frac{\vec{\mathbf{r}} \cdot (S \cdot \vec{\mathbf{r}})}{\|\vec{\mathbf{r}}\| \cdot \|S \cdot \vec{\mathbf{r}}\|} \right) \right].$$

5.2.1 Results and Discussion

As we did to check our volume-preserving proposal, we also developed a test framework to empirically study the effect of the singular values over the cage-based deformations. However, the mathematical tool introduced above does not only work with cage-based deformations, but it can also be used in any deformation performed by means of differentiable Generalised Barycentric Coordinates (GBC). As in the case of the gradient of the volume, the stretch can be computed for any methodology that defines a point $\mathbf{p}$ as the linear combination

$$\mathbf{p} = \sum_i \mathbf{v}_i \alpha_i(\mathbf{p}).$$

Consequently, we can analyse the stretch produced by the GBC presented in Chapters 2, 3 and 4. If the chosen GBC system has a closed formula, e.g. the Mean Value Coordinates (MVC) [FKR05] or the Green Coordinates[1] (GC) [LLCO08, LL10], we can get an analytical expression for the matrix M. However, for those GBC systems that do not possess a closed expression, e.g. the Harmonic Coordinates (HC) [JMD$^+$07], the T-Celestial Coordinates or the S-Celestial Coordinates in Chapter 4, we can also compute M by means of the finite difference method [New87]. In particular, we present the complete derivation of the MVC in Annex B, Section B.3.

Our experiment consisted in simulating numerous triplets of singular values (s_1, s_2, s_3), such that $s_1 \geq s_2 \geq s_3$. We observed that the rotational stress σ induced by the deformation depends on the maximum and minimum values between these singular values, that is, s_1 and s_3. Figure 5.4 shows a plot that relates the triplet (s_1, s_2, s_3) with the maximum distortion angle, i.e. the stretch σ. Additionally, Figure 5.5 relates the rotational stress σ directly with the extreme singular values

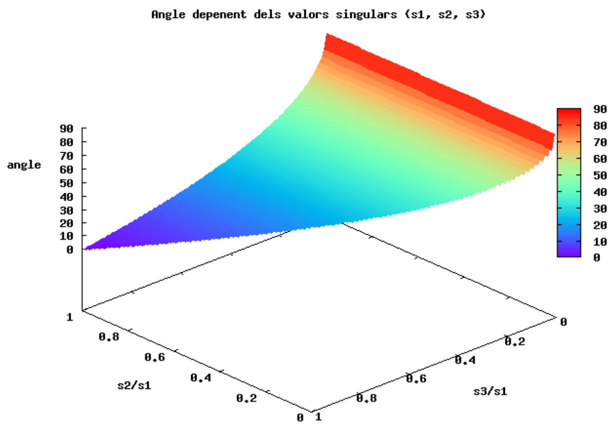


Figure 5.4: Rotational stress σ with respect to the singular values triplet (s_1, s_2, s_3).

s_1 and s_3 and, moreover, shows that the maximum angle function can be approximated with a polynomial of degree six with an absolute error of 0.44°.

Finally, the stretch measure as presented in this chapter can be used to define an algorithm to perform volume-preserving deformations driven by the modification of multiple handles at the

[1]The Green Coordinates require extra work to derive them, due to the extension of the basic linear combination with the normals of the faces of the cage.

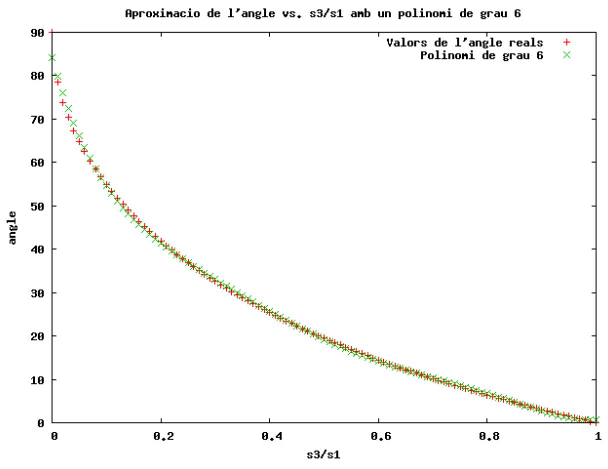


Figure 5.5: Approximation of the rotational stress σ by means of a polynomial of degree six.

same time. A preliminary idea would be to let the user move a set of control points simultaneously, without any control surface to restrict the movements. Then, the algorithm could compute how to move some of the free remaining handles in order to preserve the volume while minimizing the rotational stress produced at certain witness points $\mathbf{p}_i$, which would be located inside the volume of the object.

5.3 Summary

The constrained space deformation of 3D models is an interesting and challenging research field. In this regard, this chapter contemplates the extension of the cage-based deformations with volume-preserving capabilities.

Firstly, we introduce a technique to preserve the volume of 3D models by means of its gradient. We use this gradient as the mathematical mechanism to define a control surface to restrict the movements of the handles. Hence, when a user picks a control point, the proposed algorithm automatically computes the gradient of the volume of the model and projects the displacements made by the user over this control surface, whose normal is the gradient itself. In addition, we demonstrate that neither the gradient nor the control surface depend on the picked control point, but only on the configuration of the handles that remain unchanged. This leads us to affirm that the control surface is defined as a plane passing through the currently selected control point and with the gradient of the volume as its normal, as shown in Figure 5.1. This plane remains constant until a different control point is picked and dragged, so it only needs to be computed once each time the user selects a handle. The methodology is completely interactive, as can be seen in Table 5.2.

Although we use the Mean Value Coordinates (MVC) [FKR05] to exemplify the process, our proposal is not restricted to these Generalised Barycentric Coordinates (GBC), not even to the cage-based scheme. On the contrary, our method can be used together with any of the GBC systems described in Chapters 2, 3 and 4.

Finally, we define a measure to evaluate the local stretch, or rotational stress, caused by the deformation. We study the matrix composed of the directional derivatives of the GBC scheme of the deformation process and empirically demonstrate that the stretch depends on the maximum and minimum singular values. We also approximate the stretch function with a polynomial of degree six, as shown in Figure 5.5.

As in the previous case, the rotational stress measure is illustrated with the MVC. However, we can use it to analyse any deformation driven by a differentiable GBC system.

Chapter 6

Conclusions

"One never notices what has been done; one can only see what remains to be done."
Marie Curie

Chapter 6 Contents

The motivation behind the present PhD Dissertation is based on the purpose of developing a deformation framework that can help medical experts and related users to explore the digital representations of soft, although incompressible, organs and biological tissues. We, therefore, sought to perform intuitive, easy, flexible and interactive 3D model deformations. Additionally, we also wanted to execute these deformations in a volume-preserving manner.

Spatial deformations based on the *cage-free paradigm* provide some interesting attributes:

- **Easy set up of the handles:** the cage-free paradigm relies only on the location of the control points of the deformation. Thus, the user has to concentrate exclusively on distributing the handles around the model to be deformed. This user does not have to design any structure to connect the control points, so they can be placed wherever he or she considers more suitable to accomplish the desired deformation. The only requirement is that the region of interest must lie inside the convex hull of these control points, which is the domain of the deformation. That is, the model, or the part of the model, that the user wants to deform must be enclosed inside the convex hull of the deformation handles.

- **User-intent awareness:** this feature derives from the previous one. It refers to the fact that the user identifies important characteristics of the model during the set up of the handles and, therefore, he or she locates them accordingly.

- **Efficiency:** the pipeline of the cage-free paradigm, the same as the cage-base paradigm, consists of two steps. The first one, the set up, computes and stores the coordinates that relate the points of the model with the deformation handles. The second one, the deformation, recomputes the new location of the model points every time that a control point is dragged. This recomputation is usually done with the help of a simple and elegant linear combination of the control points by means of the previously obtained coordinates. Hence, the deformation consists of simple, easy and fast mathematical operations, which leads to an efficient and extremely parallelisable process. Any interaction of the user gets an immediate response from the system.

These features are consistent with what we were looking for in our deformation framework. For this reason, we chose to focus our efforts on studing how to produce cage-free deformations that can be applied in a medical context. Hence, this Thesis presents a compilation of different deformation techniques that consist in the description of new Generalised Barycentric Coordinate systems specifically conceived to work in this structure-free environment. Furthermore, this Dissertation also specifies a methodology to constrain any kind of spatial deformation so that the volume of the modified model remains constant.

Cage-Free Approaches

Chapter 3 describes a collection of methodologies to define new coordinate systems which, to our disappointment, do not completely meet our expectations. They fail by being discontinuous, rough or inefficient which, in turn, involves physically unplausible, visually unpleasant and incorrect deformations. The research invested in the development of these unsuccessful attempts made us realise the complexity of the underlying mathematical machinery. The discovery of new and valid coordinate systems, therefore, is not an easy task. The approaches exposed in Chapter 3 are divided into three groups: the *Mean Value Coordiantes Based Methods*, the *Layered Based Methods* and the *n-Dimensional Space Based Methods*.

The Mean Value Coordinates Based Methods define different systems of coordinates that try to take advantage of the Mean Value Coordinates [FKR05] (MVC) benefits while substituting the fixed, user-defined cage by an automatically computed connection between the control points. On the one hand, this imposed connectivity depends on the point of view of every particular model point $\mathbf{p}$, so it is not constant for all the points of the model. On the other hand, the MVC strongly depend on the connection between the deformation handles, so that two different connections result in two completely different sets of coordinates. Hence, the main problem of the Mean Value Coordinates Based Methods, although not the only one, is the discontinuities in their coordinate functions.

The Layered Based Methods substitute the classic cage by a layered structure that is used for all the points of the model. Therefore, the supporting structure remains static and immutable and the coordinates are continuous. However, the described Layered Based Methods are not smooth or local enough. These problems make them unintuitive and, moreover, the achieved deformations can present sharp features and unpleasant results.

Finally, the n-Dimensional Space Based Methods consider the problem from the n-dimensional point of view, where n is the number of user-defined control points, and they do not define any supporting structure. Instead, they directly solve systems of equations defined over the handles. However, the described methods do not always achieve good results, but they are affected by discontinuities, sharp coodinate functions and negative values.

Although none of the cage-free approaches presented in Chapter 3 give satisfactory results, the n-Dimensional Space Based Methods open a new perspective of treating the problem of building a well-defined system of coordinates. Chapter 4 exploits this new point of view to develop three new systems of coordinates that belong to the *Celestial Coordinates* family and, more precisely, to the *Point-Based Celestial Coordinates* subfamily. The Celestial Coordinates family comprises the positive Generalised Barycentric Coordinate systems that compute the coordinates of a particular model point $\mathbf{p}$ with respect to the control points by means of the projection of these handles over the *celestial sphere* of $\mathbf{p}$. In particular, the Point-Based Celestial Coordinates only aggregate those schemes in the Celestial Coordinates family that do not need any kind of connectivity between the deformation handles.

The new proposed systems in Chapter 4 are the *C-Celestial Coordinates*, the *T-Celestial Coordinates* and the *S-Celestial Coordinates.* All of them are well-defined systems of coordinates, positive, continuous and smooth. However, they suffer from global impact, i.e. the modification of any control point leads to a deformation that affects all the points inside the convex hull of these handles. For this reason, we briefly present a technique to localise the deformation process. This method consists in classifying the control points into different groups to offer user-intended shape-awareness. In this way, only those parts of the model lying inside the convex hull of a particular group will be deformed by the handles belonging to it.

Volume-Preserving Deformations

In the medical context, training applications involve the analysis of soft but incompressible organs and tissues. Therefore, the deformations must be constrained so that the volume of the modified 3D model remains constant. Chapter 5 is devoted to analysing the behaviour of the volume of the model during its manipulation process and to mathematically relating it with the displacement of the deformation handles.

We propose an algorithm to produce volume-preserving spatial deformations by means of the gradient of the volume. Every time a control point is selected, the method defines a control surface, whose normal is the gradient itself. The displacements of the handle are restricted to this control surface and, therefore, the volume of the model remains unchanged. Although the technique is exemplified with a cage-based deformation framework that uses the Mean Value Coordinates, it also works with all the Generalised Barycentric Coordinate systems presented in Section 2.3 and Chapters 3 and 4. The only exception are the Green Coordinates [LLCO08, LL10], whose gradient of the volume would need to be adapted to take into account their special linear combination.

6.1 Goals Achieved

If we compare the achieved results with the expected goals detailed in Section 1.1, we can conclude:

- **Locality:** unfortunately, all the coordinate systems and approaches presented in Chapters 3 and 4 inherit the global impact of the spatial deformations. That is, the alteration that the user induces over a certain region of the model also affects further areas inside the deformation domain. However, the additional localisation technique proposed in Section 4.5 can be a promising way to solve this problem.

- **Cage-free approach:** all the coordinate systems presented in this PhD Dissertation and, in particular, the C-Celestial Coordinates, the T-Celestial Coordinates and the S-Celestial

Coordinates have been specifically designed to work in a cage-free environment.

- **Robustness:** the Celestial Coordinates family presented in Chapter 4 give coherent and stable results under the same initial conditions, that is, the same model with the same control points surrounding it. Additionally, the technique proposed in Chapter 4 to localise the deformation and the volume-preserving method in Chapter 5 are deterministic algorithms. Therefore, given the same input, a particular model with a particular configuration of the handles, these methodologies will return the same output.

- **Real-time:** all the functional algorithms proposed in this Dissertation are efficient and ensure real-time response. The first one, the cage-free paradigm based on the proposed Celestial Coordinates, performs real-time spatial deformations. Moreover, the computation of the coordinates takes place during the set-up phase and does not have repercussion over the response of the deformation process. The second one, the localisation strategy, does not increment neither the complexity nor the mathematical operations of the deformation process. It does not, therefore, have any impact on the efficiency of the system. Finally, the volume-preserving technique computes the gradient of the volume every time a different control point is selected to be dragged. However, this operation is completely interactive and, therefore, does not decrease the response of the system.

- **Plausibility:** the combination of the Celestial Coordinates with the localisation procedure and the volume-preserving technique results in a deformation framework that ensures intuitive, plausible and pleasant transformations of elastic and incompressible bodies.

We can conclude, then, that the techniques presented in Chapters 4 and 5 fulfil the initial requirements of this research in an acceptable way. They include new deformation algorithms that ensure plausible and pleasant transformations of elastic and incompressible bodies.

6.2 Future Work

During our research, we have ventured into the *cage-free spatial deformation* domain, which had been opened by important works such as [LLD+10] and [JBPS11]. From now on, we want to step forward and improve the presented techniques, especially those detailed in Chapter 4, in five main directions. The first one refers to the local effect of the *Celestial Coordinates*. The second one aims to study the flexibility in adding and removing control points and its impact on the interactivity of the system. The third issue focuses on the study of the dimension-independence of the presented coordinate systems. The fourth extension would adjust the volume-preserving framework described in Chapter 5 to the Celestial Coordinates in Chapter 4. Finally, the fifth point concerns the study of the interaction between the user and the proposed deformation system. These five points are presented briefly below.

Increase the Local Effect

As stated before, the coordinate systems and approaches presented in Chapters 3 and 4 inherit the global impact of the spatial deformations. In Section 4.5 we have proposed a localisation technique as a promising scheme to improve this problem. We want to enhance this strategy on two main fronts. First, we want to smooth the coordinates in the *Swapping Zones*. Second, we would like to polish the interactivity of the system to make it easier to define the *Regions*.

Addition and Removal of Control Points

The classic process of spatial deformations and, in particular, the cage-based deformations defines a static cage around the model to be deformed. The coordinates to relate the points of the model with respect to the control points of the cage are then computed during a preprocess step. Finally, the user deforms the model when he or she drags the control points to a new position. This scheme, however, does not consider the modification of the initial cage. That means, if the user wants to add or remove any control point in the middle of the deformation process, the new coordinates need to be computed from scratch. Hence, the users normally adjust the cage during the *rest-pose*, that is, before the coordinates are obtained.

The approaches presented in this Dissertation suffer from the same disadvantage. However, the T-Celestial Coordinates and the S-Celestial Coordinates are computed iteratively by means of the Newton Method. Moreover, their computation is very efficient. We want to take advantatge of these two factors and analyse the costs of modifying the configuration of the handles during the deformation process. That is, we want to study if we can add or remove control points efficiently at any time during the deformation step. Our hypothesis is that, if every time that a control point is removed or added we initialise the Newton Method with the current coordinates, the iterative process that computes the new set of weights will converge quickly. Thus, the computation of the new coordiantes will be very fast and completely transparent to the user.

Dimension-Independence

The derivations of the T-Celestial Coordinates and the S-Celestial Coordinates do not depend on the 3D space. We think that they can be extended easily to any arbitrary dimension and, therefore, we want to study their dimension-independence.

Volume-Preserving Deformations and Celestial Coordinates

At this point, we simply want to integrate the T-Celestial Coordinates and the S-Celestial Coordinates into the scheme to perform volume-preserving space deformations in Chapter 5. We also want to adjust the computation of the local deformation stress to the newly proposed systems.

Interaction

Finally, we plan to study different methodologies to enrich the interaction between the users and our deformation framework. Our goal is to develop a user-friendly, intuitive, easy and convenient deformation system that can be used by users of any level of expertise in animation and deformation processes. We first want to simplify the creation of the regions used to localise the coordinates, as stated before. Then, we want to study a method to select a bunch of handles easily and modify their location together. This modification can be done by dragging the points or by rotating them around an axis defined by the user. Finally, we want to provide a procedure to configure the handles easily around the model to be deformed. The same technique can be used later to add and remove control points during the deformation step.

Appendix A

Projection over the n-Dimensional Lineal Subspace $V^{\perp}$

This Appendix is an extension to support the Chapters 3 and 4. It describes the mathematical development to compute the matrix $P_{V^{\perp}}$, which is used to project a given point $p \in \mathbb{R}^n$ to the n-dimensional lineal subspace $V^{\perp}$.

A.1 Nomenclature Aspects

Before we can proceed to mathematically develop the projection matrix, let us review the nomenclature related to $V^{\perp}$ that has been introduced in Section 3.3.

Given the control points $\vec{\mathbf{q}}_i$ projected over the unit sphere $\mathcal{S}_{\mathbf{p}}$, we want to find a vector $\boldsymbol{\beta}$ such that

$$0 = \sum_{i=1}^{n} \beta_i q_{i_x}, \qquad 0 = \sum_{i=1}^{n} \beta_i q_{i_y}, \qquad 0 = \sum_{i=1}^{n} \beta_i q_{i_z}. \tag{A.1}$$

Equation (3.16) defines the vectors χ, y and z as follows

$$\chi = \begin{pmatrix} q_{1_x} & \dots & q_{n_x} \end{pmatrix}, \quad y = \begin{pmatrix} q_{1_y} & \dots & q_{n_y} \end{pmatrix}, \quad z = \begin{pmatrix} q_{1_z} & \dots & q_{n_z} \end{pmatrix}.$$

Therefore, we can state that the vector $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 & \dots & \beta_n \end{pmatrix}$ has to be perpendicular to χ, y and z in order to solve the System (A.1), as shown in Equation (3.17).

Apart from this, the lineal subspace $V^{\perp} \subset \mathbb{R}^n$ is defined in Equation (3.18) as the subspace orthogonal to the control points

$$V^{\perp} = \{\gamma : \gamma \cdot \chi = 0 \wedge \gamma \cdot y = 0 \wedge \gamma \cdot z = 0\}. \tag{A.2}$$

Hence, $\boldsymbol{\beta} \in V^{\perp}$.

Finally, we name $\bar{p}$ the projection of the point $p \in \mathbb{R}^n$ over $V^{\perp}$, so $\bar{p} \in V^{\perp}$ and $\bar{p} \cdot \chi = \bar{p} \cdot y = \bar{p} \cdot z = 0$.

A.2 Projection Matrix $P_{V^{\perp}}$

Given a point $p \in \mathbb{R}^n$, we want to compute its projection $\bar{p}$ over the lineal subspace $V^{\perp}$, $\bar{p} \in V^{\perp}$. In other words, we want to find a matrix $P_{V^{\perp}}$ such that

$$\bar{p} = p \cdot P_{V^{\perp}}.$$

Notice that the vector $(p - \bar{p}) \in \mathbb{R}^n$ can be expressed as a convex combination of the vectors χ, y and z

$$(p - \bar{p}) = \rho_\chi \chi + \rho_y y + \rho_z z. \tag{A.3}$$

Let us define the matrix $V = \begin{pmatrix} \chi \\ y \\ z \end{pmatrix}$ and the vector $\boldsymbol{\rho} = (\rho_\chi, \rho_y, \rho_z)$, so

$$(p - \bar{p}) = \boldsymbol{\rho}^\top \cdot V. \tag{A.4}$$

By Definition (A.2) it is known that $\bar{p} \cdot V^\top = \mathbf{0}$. We, therefore, multiply Equation (A.3) by χ, y and z respectively

$$\begin{aligned} \chi \cdot (p - \bar{p}) &= \chi \cdot (\rho_\chi \chi + \rho_y y + \rho_z z) \\ y \cdot (p - \bar{p}) &= y \cdot (\rho_\chi \chi + \rho_y y + \rho_z z) \\ z \cdot (p - \bar{p}) &= z \cdot (\rho_\chi \chi + \rho_y y + \rho_z z). \end{aligned}$$

Consequently,

$$\begin{aligned} \chi \cdot p &= \chi \cdot (\rho_\chi \chi + \rho_y y + \rho_z z) \\ y \cdot p &= y \cdot (\rho_\chi \chi + \rho_y y + \rho_z z) \\ z \cdot p &= z \cdot (\rho_\chi \chi + \rho_y y + \rho_z z). \end{aligned}$$

This three equations can be rewritten in matrix notation

$$\underbrace{\begin{pmatrix} \chi \cdot \chi & \chi \cdot y & \chi \cdot z \\ \chi \cdot y & y \cdot y & y \cdot z \\ \chi \cdot z & y \cdot z & z \cdot z \end{pmatrix}}_{M} \cdot \underbrace{\begin{pmatrix} \rho_\chi \\ \rho_y \\ \rho_z \end{pmatrix}}_{\boldsymbol{\rho}^\top} = \underbrace{\begin{pmatrix} \chi \cdot p \\ y \cdot p \\ z \cdot p \end{pmatrix}}_{V \cdot p^\top}$$

and

$$\boldsymbol{\rho}^\top = M^{-1} \cdot V \cdot p^\top = p \cdot (M^{-1} \cdot V)^\top. \tag{A.5}$$

Then, we replace Equation (A.5) into Equation (A.4) to get

$$\begin{aligned} \bar{p} &= p - p \cdot (M^{-1} \cdot V)^\top \cdot V \\ &= p \cdot \left(I - V^\top \cdot (M^{-1})^\top \cdot V \right) \\ &= p \cdot \left(I - V^\top \cdot (M^\top)^{-1} \cdot V \right). \end{aligned}$$

The matrix M is symmetric, so $M^\top = M$ and

$$\bar{p} = p \cdot \left(I - V^\top \cdot M^{-1} \cdot V \right).$$

Therefore, the projection matrix $P_{V^\perp}$ is

$$P_{V^\perp} = I - V^\top \cdot M^{-1} \cdot V.$$

Appendix B

Volume-Preserving Mathematical Development

This Appendix is devoted to present the complete development and derivation of the mathematical equations involved in the volume-preserving deformations. Thus, it can be seen as an extension to support the proposals detailed in Chapter 5.

B.1 Derivation of the Gradient of the Volume

According to Section 5.1.1 and Equation (5.5), the mathematical expression to compute the volume of a 3D model represented as a triangular mesh is

$$V = \frac{1}{6}\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_j \cdot \left(\sum_{k=1}^{l}\phi_k p_k\right)\right],$$

where m is the number of triangles of the mesh, $\vec{\mathbf{N}}_j$ is the outward normal of the j-th triangle, l is the number of gaussian integration points, p_k is the k-th integration point over the triangle and ϕ_k is its corresponding gaussian weight. Then, the gradient of the volume with respect to a particular handle $\mathbf{v}_q$ corresponds to the partial derivatives

$$(\nabla V)_{\mathbf{v}_q} = \left(\frac{\partial V}{\partial v_{q_x}}, \frac{\partial V}{\partial v_{q_y}}, \frac{\partial V}{\partial v_{q_z}}\right).$$

This expands in the following three equations

$$\begin{aligned}
\frac{\partial V}{\partial v_{q_x}} &= \frac{1}{6}\left(\sum_{j=1}^{m}\left[\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_x}} \cdot \left(\sum_{k=1}^{l}\phi_k p_k\right)\right] + \sum_{j=1}^{m}\left[\vec{\mathbf{N}}_j \cdot \left(\sum_{k=1}^{l}\phi_k \frac{\partial p_k}{\partial v_{q_x}}\right)\right]\right)\\
\frac{\partial V}{\partial v_{q_y}} &= \frac{1}{6}\left(\sum_{j=1}^{m}\left[\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_y}} \cdot \left(\sum_{k=1}^{l}\phi_k p_k\right)\right] + \sum_{j=1}^{m}\left[\vec{\mathbf{N}}_j \cdot \left(\sum_{k=1}^{l}\phi_k \frac{\partial p_k}{\partial v_{q_y}}\right)\right]\right)\\
\frac{\partial V}{\partial v_{q_z}} &= \frac{1}{6}\left(\sum_{j=1}^{m}\left[\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_z}} \cdot \left(\sum_{k=1}^{l}\phi_k p_k\right)\right] + \sum_{j=1}^{m}\left[\vec{\mathbf{N}}_j \cdot \left(\sum_{k=1}^{l}\phi_k \frac{\partial p_k}{\partial v_{q_z}}\right)\right]\right).
\end{aligned}$$

Or, more compactly

$$(\nabla V)_{\mathbf{v}_q} = \frac{1}{6}\left(\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_q} \cdot \left(\sum_{k=1}^{l}\phi_k p_k\right)\right] + \sum_{j=1}^{m}\left[\vec{\mathbf{N}}_j \cdot \left(\sum_{k=1}^{l}\phi_k (\nabla p_k)_{\mathbf{v}_q}\right)\right]\right).$$

Therefore, we are interested in completely developing the gradients of the normals, $(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_q}$, and the gaussian integration points, $(\nabla p_k)_{\mathbf{v}_q}$, that is, the partial derivatives

$$(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_q} = \left(\frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_x}}, \frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_y}}, \frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_z}}\right) \quad \text{and} \quad (\nabla p_k)_{\mathbf{v}_q} = \left(\frac{\partial p_k}{\partial v_{q_x}}, \frac{\partial p_k}{\partial v_{q_y}}, \frac{\partial p_k}{\partial v_{q_z}}\right).$$

These mathematical derivations are exposed in the following sections.

B.1.1 Derivation of the Gradient $(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_q}$

First of all, we need to define the outward normal $\vec{\mathbf{N}}j$ with respect to the vertices $(\mathbf{p}_{j_1}, \mathbf{p}_{j_2}, \mathbf{p}_{j_3})$ of the j-th triangular face of the 3D model. We assume that the vertices are listed in counterclockwise order, hence, the normal $\vec{\mathbf{N}}j$ is

$$\begin{aligned}\vec{\mathbf{N}}_j &= (\mathbf{p}_{j_2} - \mathbf{p}_{j_1}) \times (\mathbf{p}_{j_3} - \mathbf{p}_{j_1}) \\ &= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ (p_{j_{2x}} - p_{j_{1x}}) & (p_{j_{2y}} - p_{j_{1y}}) & (p_{j_{2z}} - p_{j_{1z}}) \\ (p_{j_{3x}} - p_{j_{1x}}) & (p_{j_{3y}} - p_{j_{1y}}) & (p_{j_{3z}} - p_{j_{1z}}) \end{vmatrix}.\end{aligned} \tag{B.1}$$

Additionally, we know that the vertices $\mathbf{p}$ of the model are defined through a Generalized Barycentric Coordinate (GBC) system, within the context of space deformations. Thus, they can be expressed as the linear combination

$$\mathbf{p} = \sum_i \mathbf{v}_i \alpha_i(\mathbf{p}), \tag{B.2}$$

where $\mathbf{v}_i$ are the control points of the deformation. Then, we can replace Equation (B.2) into Equation (B.1), so the complete expression for the normal $\vec{\mathbf{N}}j$ is

$$\vec{\mathbf{N}}_j = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \sum_i v_{i_x}(\alpha_{i_2} - \alpha_{i_1}) & \sum_i v_{i_y}(\alpha_{i_2} - \alpha_{i_1}) & \sum_i v_{i_z}(\alpha_{i_2} - \alpha_{i_1}) \\ \sum_i v_{i_x}(\alpha_{i_3} - \alpha_{i_1}) & \sum_i v_{i_y}(\alpha_{i_3} - \alpha_{i_1}) & \sum_i v_{i_z}(\alpha_{i_3} - \alpha_{i_1}) \end{vmatrix},$$

where we substitute the GBC notation $(\alpha_i(\mathbf{p}_{j_1}), \alpha_i(\mathbf{p}_{j_2}), \alpha_i(\mathbf{p}_{j_3}))$ for $(\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3})$, for the sake of simplicity. Finally, the components of the normal vector $\vec{\mathbf{N}}_j$ are

$$\begin{aligned} N_{j_x} &= \sum_i v_{i_y}(\alpha_{i_2} - \alpha_{i_1}) \sum_i v_{i_z}(\alpha_{i_3} - \alpha_{i_1}) - \sum_i v_{i_y}(\alpha_{i_3} - \alpha_{i_1}) \sum_i v_{i_z}(\alpha_{i_2} - \alpha_{i_1}) \\ N_{j_y} &= \sum_i v_{i_z}(\alpha_{i_2} - \alpha_{i_1}) \sum_i v_{i_x}(\alpha_{i_3} - \alpha_{i_1}) - \sum_i v_{i_x}(\alpha_{i_2} - \alpha_{i_1}) \sum_i v_{i_z}(\alpha_{i_3} - \alpha_{i_1}) \\ N_{j_z} &= \sum_i v_{i_x}(\alpha_{i_2} - \alpha_{i_1}) \sum_i v_{i_y}(\alpha_{i_3} - \alpha_{i_1}) - \sum_i v_{i_y}(\alpha_{i_2} - \alpha_{i_1}) \sum_i v_{i_x}(\alpha_{i_3} - \alpha_{i_1}). \end{aligned} \tag{B.3}$$

At this point, we have a mathematical equation that relates the normal $\vec{\mathbf{N}}_j$ with the deformation handles $\mathbf{v}_i$, so we are able to expand the gradient $(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_q} = \left(\frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_x}}, \frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_y}}, \frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_z}}\right)$. Then, the components of this gradient are the following

$$\begin{aligned}
\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_x}} &= \left(\frac{\partial N_{j_x}}{\partial v_{q_x}}, \frac{\partial N_{j_y}}{\partial v_{q_x}}, \frac{\partial N_{j_z}}{\partial v_{q_x}}\right) \\
&= \Bigg(0, (\alpha_{q3} - \alpha_{q1}) \sum_i v_{i_z}(\alpha_{i_2} - \alpha_{i_1}) - (\alpha_{q2} - \alpha_{q1}) \sum_i v_{i_z}(\alpha_{i_3} - \alpha_{i_1}), \\
&\qquad (\alpha_{q2} - \alpha_{q1}) \sum_i v_{i_y}(\alpha_{i_3} - \alpha_{i_1}) - (\alpha_{q3} - \alpha_{q1}) \sum_i v_{i_y}(\alpha_{i_2} - \alpha_{i_1})\Bigg) \\
\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_y}} &= \left(\frac{\partial N_{j_x}}{\partial v_{q_y}}, \frac{\partial N_{j_y}}{\partial v_{q_y}}, \frac{\partial N_{j_z}}{\partial v_{q_y}}\right) \\
&= \Bigg((\alpha_{q2} - \alpha_{q1}) \sum_i v_{i_z}(\alpha_{i_3} - \alpha_{i_1}) - (\alpha_{q3} - \alpha_{q1}) \sum_i v_{i_z}(\alpha_{i_2} - \alpha_{i_1}), 0, \\
&\qquad (\alpha_{q3} - \alpha_{q1}) \sum_i v_{i_x}(\alpha_{i_2} - \alpha_{i_1}) - (\alpha_{q2} - \alpha_{q1}) \sum_i v_{i_x}(\alpha_{i_3} - \alpha_{i_1})\Bigg) \\
\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_z}} &= \left(\frac{\partial N_{j_x}}{\partial v_{q_z}}, \frac{\partial N_{j_y}}{\partial v_{q_z}}, \frac{\partial N_{j_z}}{\partial v_{q_z}}\right) \\
&= \Bigg((\alpha_{q3} - \alpha_{q1}) \sum_i v_{i_y}(\alpha_{i_2} - \alpha_{i_1}) - (\alpha_{q2} - \alpha_{q1}) \sum_i v_{i_y}(\alpha_{i_3} - \alpha_{i_1}), \\
&\qquad (\alpha_{q2} - \alpha_{q1}) \sum_i v_{i_x}(\alpha_{i_3} - \alpha_{i_1}) - (\alpha_{q3} - \alpha_{q1}) \sum_i v_{i_x}(\alpha_{i_2} - \alpha_{i_1}), 0\Bigg).
\end{aligned}$$

B.1.2 Derivation of the Gradient $(\nabla p_k)_{\mathbf{v}_q}$

According to Section 5.1.1 and Equation (5.4), the gaussian integration points p_k over a triangle j are defined as

$$p_k = \eta_{k_1}\mathbf{p}_{j_1} + \eta_{k_2}\mathbf{p}_{j_2} + \eta_{k_3}\mathbf{p}_{j_3},$$

where $(\mathbf{p}_{j_1}, \mathbf{p}_{j_2}, \mathbf{p}_{j_3})$ are the vertices of this triangle. Once again, we can define the vertices $\mathbf{p}$ of the model with the well-known linear combination

$$\mathbf{p} = \sum_i \mathbf{v}_i \alpha_i(\mathbf{p}).$$

Hence, the gaussian integration points can be expressed with respect to the deformation control points $\mathbf{v}_i$ as

$$p_k = \eta_{k_1} \sum_i \mathbf{v}_i \alpha_{i_1} + \eta_{k_2} \sum_i \mathbf{v}_i \alpha_{i_2} + \eta_{k_3} \sum_i \mathbf{v}_i \alpha_{i_3}, \tag{B.4}$$

where we substitute the GBC notation $(\alpha_i(\mathbf{p}_{j_1}), \alpha_i(\mathbf{p}_{j_2}), \alpha_i(\mathbf{p}_{j_3}))$ for $(\alpha_{i_1}, \alpha_{i_2}, \alpha_{i_3})$, as done in the previous section. Then, components of the gradient $(\nabla p_k)_{\mathbf{v}_q} = \left(\frac{\partial p_k}{\partial v_{q_x}}, \frac{\partial p_k}{\partial v_{q_y}}, \frac{\partial p_k}{\partial v_{q_z}}\right)$ with regard to a particular handle $\mathbf{v}_q$ are the following

$$\begin{aligned}
\frac{\partial p_k}{\partial v_{q_x}} &= \left(\frac{\partial p_{k_x}}{\partial v_{q_x}}, \frac{\partial p_{k_y}}{\partial v_{q_x}}, \frac{\partial p_{k_z}}{\partial v_{q_x}}\right) \\
&= (\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}, 0, 0) \\
\frac{\partial p_k}{\partial v_{q_y}} &= \left(\frac{\partial p_{k_x}}{\partial v_{q_y}}, \frac{\partial p_{k_y}}{\partial v_{q_y}}, \frac{\partial p_{k_z}}{\partial v_{q_y}}\right) \\
&= (0, \eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}, 0) \\
\frac{\partial p_k}{\partial v_{q_z}} &= \left(\frac{\partial p_{k_x}}{\partial v_{q_z}}, \frac{\partial p_{k_y}}{\partial v_{q_z}}, \frac{\partial p_{k_z}}{\partial v_{q_z}}\right) \\
&= (0, 0, \eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3})\,.
\end{aligned} \tag{B.5}$$

B.2 Study of the Gradient of the Volume

According to Section 5.1.3, we want to study how the gradient of the volume behaves when a particular control point $\mathbf{v}_q$ is picked and dragged an arbitrary vector $\vec{\boldsymbol{\delta}}_q$. For this reason, we split its definition into two components

$$(\nabla V)_{\mathbf{v}_q} = (\nabla V)_{\mathbf{v}_{q,q}} + (\nabla V)_{\mathbf{v}_{q,\bar{q}}},$$

where $(\nabla V)_{\mathbf{v}_{q,q}}$ clusters together the terms that contain $\mathbf{v}_q$ and $(\nabla V)_{\mathbf{v}_{q,\bar{q}}}$ the remaining ones. We know that $(\nabla V)_{\mathbf{v}_{q,\bar{q}}}$ remains constant during the translation of $\mathbf{v}_q$, i.e. it depends exclusively on the unmoving control points. Thus, we only need to analyse the behaviour of the $(\nabla V)_{\mathbf{v}_{q,q}}$ component, whose simplification process is shown below

$$\begin{aligned}
(\nabla V)_{\mathbf{v}_{q,q}} = \frac{1}{6}\Bigg[\Bigg(&\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}} \cdot \left(\sum_{k=1}^{l}\phi_k p_{k,q}\right)\right] + \sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j))_{\mathbf{v}_{q,q}} \cdot \left(\sum_{k=1}^{l}\phi_k p_{k,\bar{q}}\right)\right] + \\
&+ \sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,\bar{q}}} \cdot \left(\sum_{k=1}^{l}\phi_k p_{k,q}\right)\right]\Bigg) + \\
+ \Bigg(&\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{j,q} \cdot \left(\sum_{k=1}^{l}\phi_k (\nabla p_k)_{\mathbf{v}_{q,q}}\right)\right] + \sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{j,q} \cdot \left(\sum_{k=1}^{l}\phi_k (\nabla p_k)_{\mathbf{v}_{q,\bar{q}}}\right)\right] + \\
&+ \sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{j,\bar{q}} \cdot \left(\sum_{k=1}^{l}\phi_k (\nabla p_k)_{\mathbf{v}_{q,q}}\right)\right]\Bigg)\Bigg]\,.
\end{aligned}$$

Then,

$$(\nabla V)_{\mathbf{v}_{q,q}} = \frac{1}{6}\left[\left(\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}}\cdot\left(\sum_{k=1}^{l}\phi_k p_{k,q}+\sum_{k=1}^{l}\phi_k p_{k,\bar{q}}\right)\right]+\right.\right.$$
$$\left.\left.+\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,\bar{q}}}\cdot\left(\sum_{k=1}^{l}\phi_k p_{k,q}\right)\right]\right)\right]+$$
$$+\left(\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{j,q}\cdot\left(\sum_{k=1}^{l}\phi_k(\nabla p_k)_{\mathbf{v}_{q,q}}+\sum_{k=1}^{l}\phi_k(\nabla p_k)_{\mathbf{v}_{q,\bar{q}}}\right)\right]+\right.$$
$$\left.\left.+\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{j,\bar{q}}\cdot\left(\sum_{k=1}^{l}\phi_k(\nabla p_k)_{\mathbf{v}_{q,q}}\right)\right]\right)\right].$$

Notice that $(\nabla p_k)_{\mathbf{v}_{q,q}} = \vec{\mathbf{0}}$, as it can be derived from the set of Equations (B.5). As a consequence, $(\nabla p_k)_{\mathbf{v}_q} = (\nabla p_k)_{\mathbf{v}_{q,\bar{q}}}$. Additionally, $\left(\sum_{k=1}^{l}\phi_k p_{k,q}+\sum_{k=1}^{l}\phi_k p_{k,\bar{q}}\right) = \sum_{k=1}^{l}\phi_k p_k$ and $\left(\sum_{k=1}^{l}\phi_k(\nabla p_k)_{\mathbf{v}_{q,q}}+\sum_{k=1}^{l}\phi_k(\nabla p_k)_{\mathbf{v}_{q,\bar{q}}}\right) = \sum_{k=1}^{l}\phi_k(\nabla p_k)_{\mathbf{v}_q}$. Hence, the mathematical expression for the component $(\nabla V)_{\mathbf{v}_{q,q}}$ is

$$(\nabla V)_{\mathbf{v}_{q,q}} = \frac{1}{6}\left(\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}}\cdot\left(\sum_{k=1}^{l}\phi_k p_k\right)\right]+\right.$$
$$\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,\bar{q}}}\cdot\left(\sum_{k=1}^{l}\phi_k p_{k,q}\right)\right]+ \tag{B.6}$$
$$\left.\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{q,q}\cdot\left(\sum_{k=1}^{l}\phi_k\nabla p_k\right)\right]\right).$$

The following sections study the different terms that compose the formula for $(\nabla V)_{\mathbf{v}_{q,q}}$.

B.2.1 Study of the Gradient $(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}}$

In order to simplify the study, we analyse the coordinates of $(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}}$ separately. Thus, we first examine its component x, that is, the partial derivative $\left(\frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_x}}\right)_q$, which groups the terms that contain the control point $\mathbf{v}_q$.

The terms of the component x of the gradient $(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_q}$ that contain the handle $\mathbf{v}_q$ are the following

$$\left(\frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_x}}\right)_q = (0, (\alpha_{q_3}-\alpha_{q_1})v_{q_z}(\alpha_{q_2}-\alpha_{q_1}) - (\alpha_{q_2}-\alpha_{q_1})v_{q_z}(\alpha_{q_3}-\alpha_{q_1}),$$
$$(\alpha_{q_2}-\alpha_{q_1})v_{q_y}(\alpha_{q_3}-\alpha_{q_1}) - (\alpha_{q_3}-\alpha_{q_1})v_{q_y}(\alpha_{q_2}-\alpha_{q_1}))\,.$$

Consequently, the partial derivative $\left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_x}}\right)_q = \mathbf{0}$ is null. The same result is obtained for the components y and z, so we can affirm

$$\left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_x}}\right)_q = \left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_y}}\right)_q = \left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_z}}\right)_q = \mathbf{0} \quad \text{and} \quad (\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}} = \vec{\mathbf{0}}.$$

This fact leads to the cancellation of the term $\sum_{j=1}^{m}\left[(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}} \cdot \left(\sum_{k=1}^{l} \phi_k p_k\right)\right]$ from Equation (B.6)

$$\sum_{j=1}^{m}\left[(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}} \cdot \left(\sum_{k=1}^{l} \phi_k p_k\right)\right] = \mathbf{0}.$$

B.2.2 Study of the Expression $(\nabla \vec{\mathbf{N}}_j)_{\mathbf{v}_{q,\bar{q}}} \cdot p_{k,q}$

According to Equation (B.4), the point $p_{k,q}$, which only clusters the terms in p_k that contain $\mathbf{v}_q$, is

$$p_{k,q} = \mathbf{v}_q(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}).$$

In order to make the final evaluation of Equation (B.6) easier, we simplify the expressions $\left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_x}}\right)_{\bar{q}}$, $\left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_y}}\right)_{\bar{q}}$ and $\left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_z}}\right)_{\bar{q}}$, that group the terms that do not include the handle $\mathbf{v}_q$. Hence,

$$\begin{aligned}
\left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_x}}\right)_{\bar{q}} &= \left(0, \left(\frac{\partial \vec{\mathbf{N}}_{j_y}}{\partial v_{q_x}}\right)_{\bar{q}}, \left(\frac{\partial \vec{\mathbf{N}}_{j_z}}{\partial v_{q_x}}\right)_{\bar{q}}\right)\\
\left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_y}}\right)_{\bar{q}} &= \left(\left(\frac{\partial \vec{\mathbf{N}}_{j_y}}{\partial v_{q_y}}\right)_{\bar{q}}, 0, \left(\frac{\partial \vec{\mathbf{N}}_{j_z}}{\partial v_{q_y}}\right)_{\bar{q}}\right)\\
\left(\frac{\partial \vec{\mathbf{N}}_j}{\partial v_{q_z}}\right)_{\bar{q}} &= \left(\left(\frac{\partial \vec{\mathbf{N}}_{j_y}}{\partial v_{q_z}}\right)_{\bar{q}}, \left(\frac{\partial \vec{\mathbf{N}}_{j_z}}{\partial v_{q_z}}\right)_{\bar{q}}, 0\right).
\end{aligned}$$

Then, the components x, y and z of the expression $(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,\bar{q}}} \cdot p_{k,q}$ are

$$\begin{aligned}
\left(\frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_x}}\right)_{\bar{q}} \cdot p_{k,q} = {} & \left(\frac{\partial\vec{\mathbf{N}}_{j_y}}{\partial v_{q_x}}\right)_{\bar{q}} v_{q_y}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}) + \\
& + \left(\frac{\partial\vec{\mathbf{N}}_{j_z}}{\partial v_{q_x}}\right)_{\bar{q}} v_{q_z}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}) \\
\left(\frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_y}}\right)_{\bar{q}} \cdot p_{k,q} = {} & \left(\frac{\partial\vec{\mathbf{N}}_{j_x}}{\partial v_{q_y}}\right)_{\bar{q}} v_{q_x}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}) + \\
& + \left(\frac{\partial\vec{\mathbf{N}}_{j_z}}{\partial v_{q_y}}\right)_{\bar{q}} v_{q_z}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}) \\
\left(\frac{\partial\vec{\mathbf{N}}_j}{\partial v_{q_z}}\right)_{\bar{q}} \cdot p_{k,q} = {} & \left(\frac{\partial\vec{\mathbf{N}}_{j_x}}{\partial v_{q_z}}\right)_{\bar{q}} v_{q_x}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}) + \\
& + \left(\frac{\partial\vec{\mathbf{N}}_{j_y}}{\partial v_{q_z}}\right)_{\bar{q}} v_{q_y}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}).
\end{aligned}$$

B.2.3 Study of the Expression $\vec{\mathrm{N}}_{j,q} \cdot (\nabla p_k)_{\mathbf{v}_q}$

Similarly to previous sections, we study the coordinates of the expression $\vec{\mathbf{N}}_{j,q} \cdot (\nabla p_k)_{\mathbf{v}_q}$ separately. Then, according to the set of Equations (B.3), the coordinate x of the normal $\vec{\mathbf{N}}_{j,q}$, which groups the terms where the handle $\mathbf{v}_q$ appears, can be reorganized as

$$\begin{aligned}
\vec{\mathbf{N}}_{j_x,q} = {} & \left(v_{q_y}(\alpha_{q_2} - \alpha_{q_1}) \sum_{i \neq q} v_{i_z}(\alpha_{i_3} - \alpha_{i_1}) - v_{q_y}(\alpha_{q_3} - \alpha_{q_1}) \sum_{i \neq q} v_{i_z}(\alpha_{i_2} - \alpha_{i_1}) \right) + \\
& + \left(v_{q_z}(\alpha_{q_3} - \alpha_{q_1}) \sum_{i \neq q} v_{i_y}(\alpha_{i_2} - \alpha_{i_1}) - v_{q_z}(\alpha_{q_2} - \alpha_{q_1}) \sum_{i \neq q} v_{i_y}(\alpha_{i_3} - \alpha_{i_1}) \right).
\end{aligned}$$

Observe that $\vec{\mathbf{N}}_{j_x,q}$ can also be expressed as

$$\vec{\mathbf{N}}_{j_x,q} = -v_{q_y} \left(\frac{\partial\vec{\mathbf{N}}_{j_y}}{\partial v_{q_x}}\right)_{\bar{q}} - v_{q_z} \left(\frac{\partial\vec{\mathbf{N}}_{j_z}}{\partial v_{q_x}}\right)_{\bar{q}}.$$

Hence, when we multiply $\vec{\mathbf{N}}_{j_x,q} \cdot (\nabla p_k)_{\mathbf{v}_q}$, the obtained result is

$$\begin{aligned}
\vec{\mathbf{N}}_{j_x,q} \cdot (\nabla p_k)_{\mathbf{v}_q} = {} & -\left(\frac{\partial\vec{\mathbf{N}}_{j_y}}{\partial v_{q_x}}\right)_{\bar{q}} v_{q_y}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}) - \\
& - \left(\frac{\partial\vec{\mathbf{N}}_{j_z}}{\partial v_{q_x}}\right)_{\bar{q}} v_{q_z}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}).
\end{aligned}$$

If we apply the same mathematical derivation over the coordinates y and z, we find

$$\begin{aligned}\vec{\mathbf{N}}_{j_y,q} \cdot (\nabla p_k)_{\mathbf{v}_q} = &-\left(\frac{\partial \vec{\mathbf{N}}_{j_x}}{\partial v_{q_y}}\right)_{\bar{q}} v_{q_x}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}) - \\ &-\left(\frac{\partial \vec{\mathbf{N}}_{j_z}}{\partial v_{q_y}}\right)_{\bar{q}} v_{q_z}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}) \\ \vec{\mathbf{N}}_{j_z,q} \cdot (\nabla p_k)_{\mathbf{v}_q} = &-\left(\frac{\partial \vec{\mathbf{N}}_{j_x}}{\partial v_{q_z}}\right)_{\bar{q}} v_{q_x}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}) - \\ &-\left(\frac{\partial \vec{\mathbf{N}}_{j_y}}{\partial v_{q_z}}\right)_{\bar{q}} v_{q_y}(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3}).\end{aligned}$$

B.2.4 Final Result for the Expression $(\nabla V)_{\mathbf{v}_{q,q}}$

Finally, we can put together all the derivations exposed above and determine the influence of the component $(\nabla V)_{\mathbf{v}_{q,q}}$ over the general gradient $(\nabla V)_{\mathbf{v}_q}$. Then, from Equation (B.6), we know that

$$\begin{aligned}(\nabla V)_{\mathbf{v}_{q,q}} = \frac{1}{6}\Bigg(&\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}} \cdot \left(\sum_{k=1}^{l}\phi_k p_k\right)\right] + \\ &\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,\bar{q}}} \cdot \left(\sum_{k=1}^{l}\phi_k p_{k,q}\right)\right] + \\ &\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{q,q} \cdot \left(\sum_{k=1}^{l}\phi_k \nabla p_k\right)\right]\Bigg).\end{aligned}$$

First, Section B.2.1 has led us to the cancellation of the term $\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,q}} \cdot \left(\sum_{k=1}^{l}\phi_k p_k\right)\right]$. Then, Section B.2.2 allows us to arrange the expression $\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,\bar{q}}} \cdot \left(\sum_{k=1}^{l}\phi_k p_{k,q}\right)\right]$ as

$$\begin{aligned}&\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,\bar{q}}} \cdot \left(\sum_{k=1}^{l}\phi_k p_{k,q}\right)\right] = \\ &\quad\sum_{j=1}^{m}\Bigg[\Bigg(\left(\frac{\partial \vec{\mathbf{N}}_{j_x}}{\partial v_{q_y}}\right)_{\bar{q}} + \left(\frac{\partial \vec{\mathbf{N}}_{j_x}}{\partial v_{q_z}}\right)_{\bar{q}}, \left(\frac{\partial \vec{\mathbf{N}}_{j_y}}{\partial v_{q_x}}\right)_{\bar{q}} + \left(\frac{\partial \vec{\mathbf{N}}_{j_y}}{\partial v_{q_z}}\right)_{\bar{q}}, \left(\frac{\partial \vec{\mathbf{N}}_{j_z}}{\partial v_{q_x}}\right)_{\bar{q}} + \left(\frac{\partial \vec{\mathbf{N}}_{j_z}}{\partial v_{q_y}}\right)_{\bar{q}}\Bigg) \cdot \\ &\quad\cdot\left(\mathbf{v}_q \sum_{k=1}^{l}\phi_k(\eta_{k_1}\alpha_{q_1} + \eta_{k_2}\alpha_{q_2} + \eta_{k_3}\alpha_{q_3})\right)\Bigg]\end{aligned} \tag{B.7}$$

Finally, Section B.2.3 redefines expression $\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{q,q}\cdot\left(\sum_{k=1}^{l}\phi_k\nabla p_k\right)\right]$ as

$$\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{q,q}\cdot\left(\sum_{k=1}^{l}\phi_k\nabla p_k\right)\right]=$$
$$\sum_{j=1}^{m}\left[-\left(\left(\frac{\partial\vec{\mathbf{N}}_{j_x}}{\partial v_{q_y}}\right)_{\bar{q}}+\left(\frac{\partial\vec{\mathbf{N}}_{j_x}}{\partial v_{q_z}}\right)_{\bar{q}},\left(\frac{\partial\vec{\mathbf{N}}_{j_y}}{\partial v_{q_x}}\right)_{\bar{q}}+\left(\frac{\partial\vec{\mathbf{N}}_{j_y}}{\partial v_{q_z}}\right)_{\bar{q}},\left(\frac{\partial\vec{\mathbf{N}}_{j_z}}{\partial v_{q_x}}\right)_{\bar{q}}+\left(\frac{\partial\vec{\mathbf{N}}_{j_z}}{\partial v_{q_y}}\right)_{\bar{q}}\right)\cdot\right.$$
$$\left.\cdot\left(\mathbf{v}_q\sum_{k=1}^{l}\phi_k(\eta_{k_1}\alpha_{q_1}+\eta_{k_2}\alpha_{q_2}+\eta_{k_3}\alpha_{q_3})\right)\right] \tag{B.8}$$

Notice that Equations (B.7) and (B.8) cancel each other, so $\sum_{j=1}^{m}\left[(\nabla\vec{\mathbf{N}}_j)_{\mathbf{v}_{q,\bar{q}}}\cdot\left(\sum_{k=1}^{l}\phi_k p_{k,q}\right)\right]=\sum_{j=1}^{m}\left[\vec{\mathbf{N}}_{q,q}\cdot\left(\sum_{k=1}^{l}\phi_k\nabla p_k\right)\right]$. Then, we can conclude that the component $(\nabla V)_{\mathbf{v}_{q,q}}$ is null, that is, $(\nabla V)_{\mathbf{v}_{q,q}}=\vec{\mathbf{0}}$. Hence, the gradient $(\nabla V)_{\mathbf{v}_q}$ does not depend on $(\nabla V)_{\mathbf{v}_{q,q}}$, but it only depends on $(\nabla V)_{\mathbf{v}_{q,\bar{q}}}$. This means that the gradient of the volume is determined by the control points that remain static and it is not affected by the modification of the handle $\mathbf{v}_q$. Consequently, the gradient is kept constant since the user picks the vertex $\mathbf{v}_q$ until he/she releases it to select a different handle.

B.3 Derivation of the Mean Value Coordinates

According to Floater et al. [FKR05] and Equations (3.7) and (3.8) from Section 3.1, Chapter 3, the Mean Value Coordinate (MVC) of the point $\mathbf{p}$ with respect to a control point $\mathbf{v}_i$ is defined by the equations below

$$\begin{aligned}\alpha_i(\mathbf{p}) &= \frac{\omega_i(\mathbf{p})}{\sum_{j=1}^{n}\omega_j(\mathbf{p})}\\ \omega_i(\mathbf{p}) &= \frac{1}{d_i}\sum_{T\ni\mathbf{v_i}}\frac{\gamma_{jk}+\gamma_{ij}\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk}+\gamma_{ki}\vec{\mathbf{n}}_{ki}\cdot\vec{\mathbf{n}}_{jk}}{2\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk}}.\end{aligned}$$

The following sections present the partial derivatives of $\alpha_i(\mathbf{p})$ with respect to $\mathbf{p}$. They can be used in Section 5.2 to evaluate the stretch induced to any model subjected to a cage-based deformation driven by the MVC.

B.3.1 Derivation of the Expressions $(\nabla\gamma_{ij})_\mathbf{p}$, $(\nabla\gamma_{jk})_\mathbf{p}$ and $(\nabla\gamma_{ki})_\mathbf{p}$

In this section, we compute the gradient $(\nabla\gamma_{ij})_\mathbf{p}$ as an example. The other two gradients can be easily obtained with the same mathematical reasoning. Moreover, $(\nabla\gamma_{jk})_\mathbf{p}$ and $(\nabla\gamma_{ki})_\mathbf{p}$ can be directly derived from the definition of $(\nabla\gamma_{ij})_\mathbf{p}$ by replacing the sub-indices (i,j) by (j,k) and (k,i), respectively.

First of all, we present the complete expression for the angle γ_{ij} as defined in Section 3.1

$$\begin{aligned}\gamma_{ij} &= \arccos\left(\frac{(\mathbf{v}_i-\mathbf{p})\cdot(\mathbf{v}_j-\mathbf{p})}{\|(\mathbf{v}_i-\mathbf{p})\|\|(\mathbf{v}_j-\mathbf{p})\|}\right)\\ &= \arccos\left(\frac{(v_{i_x}-p_x)(v_{j_x}-p_x)+(v_{i_y}-p_y)(v_{j_y}-p_y)+(v_{i_z}-p_z)(v_{j_z}-p_z)}{\sqrt{(v_{i_x}-p_x)^2+(v_{i_y}-p_y)^2+(v_{i_z}-p_z)^2}\sqrt{(v_{j_x}-p_x)^2+(v_{j_y}-p_y)^2+(v_{j_z}-p_z)^2}}\right)\\ &= \arccos(\mathcal{U})\end{aligned}$$

and rename $\mathcal{U}=\frac{(v_{i_x}-p_x)(v_{j_x}-p_x)+(v_{i_y}-p_y)(v_{j_y}-p_y)+(v_{i_z}-p_z)(v_{j_z}-p_z)}{\sqrt{(v_{i_x}-p_x)^2+(v_{i_y}-p_y)^2+(v_{i_z}-p_z)^2}\sqrt{(v_{j_x}-p_x)^2+(v_{j_y}-p_y)^2+(v_{j_z}-p_z)^2}}$ to simplify the following development.

Then, the gradient $(\nabla\gamma_{ij})_\mathbf{p}$ is

$$(\nabla\gamma_{ij})_\mathbf{p}=\left(\frac{\partial\gamma_{ij}}{\partial p_x},\frac{\partial\gamma_{ij}}{\partial p_y},\frac{\partial\gamma_{ij}}{\partial p_z}\right),$$

where

$$\frac{\partial\gamma_{ij}}{\partial p_x}=-\frac{\frac{\partial\mathcal{U}}{\partial p_x}}{\sqrt{1-\mathcal{U}^2}}\quad\text{and}\quad\frac{\partial\gamma_{ij}}{\partial p_y}=-\frac{\frac{\partial\mathcal{U}}{\partial p_y}}{\sqrt{1-\mathcal{U}^2}}\quad\text{and}\quad\frac{\partial\gamma_{ij}}{\partial p_z}=-\frac{\frac{\partial\mathcal{U}}{\partial p_z}}{\sqrt{1-\mathcal{U}^2}}.$$

Finally, the partial derivatives $\frac{\partial\mathcal{U}}{\partial p_x}$, $\frac{\partial\mathcal{U}}{\partial p_y}$ and $\frac{\partial\mathcal{U}}{\partial p_z}$ are shown next

$$\begin{aligned}\frac{\partial\mathcal{U}}{\partial p_x} &= \frac{(-v_{i_x}-v_{j_x}+2p_x)\|\vec{\mathbf{pv}}_i\|\|\vec{\mathbf{pv}}_j\|+\left(\frac{(v_{i_x}-p_x)}{\|\vec{\mathbf{pv}}_i\|}\|\vec{\mathbf{pv}}_j\|+\frac{(v_{j_x}-p_x)}{\|\vec{\mathbf{pv}}_j\|}\|\vec{\mathbf{pv}}_i\|\right)\vec{\mathbf{pv}}_i\cdot\vec{\mathbf{pv}}_j}{(\|\vec{\mathbf{pv}}_i\|\|\vec{\mathbf{pv}}_j\|)^2}\\ \frac{\partial\mathcal{U}}{\partial p_y} &= \frac{(-v_{i_y}-v_{j_y}+2p_y)\|\vec{\mathbf{pv}}_i\|\|\vec{\mathbf{pv}}_j\|+\left(\frac{(v_{i_y}-p_y)}{\|\vec{\mathbf{pv}}_i\|}\|\vec{\mathbf{pv}}_j\|+\frac{(v_{j_y}-p_y)}{\|\vec{\mathbf{pv}}_j\|}\|\vec{\mathbf{pv}}_i\|\right)\vec{\mathbf{pv}}_i\cdot\vec{\mathbf{pv}}_j}{(\|\vec{\mathbf{pv}}_i\|\|\vec{\mathbf{pv}}_j\|)^2}\\ \frac{\partial\mathcal{U}}{\partial p_z} &= \frac{(-v_{i_z}-v_{j_z}+2p_z)\|\vec{\mathbf{pv}}_i\|\|\vec{\mathbf{pv}}_j\|+\left(\frac{(v_{i_z}-p_z)}{\|\vec{\mathbf{pv}}_i\|}\|\vec{\mathbf{pv}}_j\|+\frac{(v_{j_z}-p_z)}{\|\vec{\mathbf{pv}}_j\|}\|\vec{\mathbf{pv}}_i\|\right)\vec{\mathbf{pv}}_i\cdot\vec{\mathbf{pv}}_j}{(\|\vec{\mathbf{pv}}_i\|\|\vec{\mathbf{pv}}_j\|)^2}.\end{aligned}$$

B.3.2 Derivation of the Expressions $(\nabla(\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk}))_\mathbf{p}$ and $(\nabla(\vec{\mathbf{n}}_{ki}\cdot\vec{\mathbf{n}}_{jk}))_\mathbf{p}$

Similarly to the previous section, we compute the gradient $(\nabla(\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk}))_\mathbf{p}$ as an example. The remaining one can be computed by the same procedure.

The normals $\vec{\mathbf{n}}_{ij}$, $\vec{\mathbf{n}}_{jk}$ and $\vec{\mathbf{n}}_{ki}$ are unitary and are defined by the point $\mathbf{p}$ and the control points $\mathbf{v}_i$,

$\mathbf{v}_j$ and $\mathbf{v}_k$. Thus, we first develop the formula for $\vec{\mathbf{n}}_{ij}$ as an exemplification

$$
\begin{aligned}
\vec{\mathbf{N}}_{ij} &= (\mathbf{v}_i - \mathbf{p}) \times (\mathbf{v}_j - \mathbf{p}) \\
&= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ v_{i_x} - p_x & v_{i_y} - p_y & v_{i_z} - p_z \\ v_{j_x} - p_x & v_{j_y} - p_y & v_{j_z} - p_z \end{vmatrix} \\
&= \big((v_{i_y} - p_y)(v_{j_z} - p_z) - (v_{i_z} - p_z)(v_{j_y} - p_y), (v_{i_z} - p_z)(v_{j_x} - p_x) - (v_{i_x} - p_x)(v_{j_z} - p_z), \\
&\qquad (v_{i_x} - p_x)(v_{j_y} - p_y) - (v_{i_y} - p_y)(v_{j_x} - p_x)\big) \\
\vec{\mathbf{n}}_{ij} &= \frac{\vec{\mathbf{N}}_{ij}}{\|\vec{\mathbf{N}}_{ij}\|}.
\end{aligned}
\tag{B.9}
$$

Then the scalar product $\vec{\mathbf{n}}_{ij} \cdot \vec{\mathbf{n}}_{jk}$ corresponds to the following expression

$$
\begin{aligned}
\vec{\mathbf{N}}_{ij} \cdot \vec{\mathbf{N}}_{jk} = &\big[(v_{i_y} - p_y)(v_{j_z} - p_z) - (v_{i_z} - p_z)(v_{j_y} - p_y)\big] \cdot \\
&\cdot \big[(v_{j_y} - p_y)(v_{k_z} - p_z) - (v_{j_z} - p_z)(v_{k_y} - p_y)\big] + \\
&+ [(v_{i_z} - p_z)(v_{j_x} - p_x) - (v_{i_x} - p_x)(v_{j_z} - p_z)] \cdot \\
&\cdot [(v_{j_z} - p_z)(v_{k_x} - p_x) - (v_{j_x} - p_x)(v_{k_z} - p_z)] + \\
&+ \big[(v_{i_x} - p_x)(v_{j_y} - p_y) - v_{i_y} - p_y)(v_{j_x} - p_x)\big] \cdot \\
&\cdot \big[(v_{j_x} - p_x)(v_{k_y} - p_y) - (v_{j_y} - p_y)(v_{k_x} - p_x)\big] \\
\vec{\mathbf{n}}_{ij} \cdot \vec{\mathbf{n}}_{jk} = &\frac{\vec{\mathbf{N}}_{ij} \cdot \vec{\mathbf{N}}_{jk}}{\|\vec{\mathbf{N}}_{ij}\|\|\vec{\mathbf{N}}_{jk}\|}.
\end{aligned}
$$

Finally, the gradient $(\nabla(\vec{\mathbf{n}}_{ij} \cdot \vec{\mathbf{n}}_{jk}))_{\mathbf{p}}$ is

$$
(\nabla(\vec{\mathbf{n}}_{ij} \cdot \vec{\mathbf{n}}_{jk}))_{\mathbf{p}} = \left(\frac{\partial(\vec{\mathbf{n}}_{ij} \cdot \vec{\mathbf{n}}_{jk})}{\partial p_x}, \frac{\partial(\vec{\mathbf{n}}_{ij} \cdot \vec{\mathbf{n}}_{jk})}{\partial p_y}, \frac{\partial(\vec{\mathbf{n}}_{ij} \cdot \vec{\mathbf{n}}_{jk})}{\partial p_z} \right),
$$

where

$$\begin{aligned}
\frac{\partial(\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk})}{\partial p_x} =& \Bigg(\big[(v_{j_z}-v_{i_z})n_{jk_y}+(v_{k_z}-v_{j_z})n_{ij_y}+ \\
&+(v_{i_y}-v_{j_y})n_{jk_z}+(v_{j_y}-v_{k_y})n_{ij_z}\big]\,\|\vec{\mathbf{N}}_{ij}\|\|\vec{\mathbf{N}}_{jk}\|- \\
&-\vec{\mathbf{N}}_{ij}\cdot\vec{\mathbf{N}}_{jk}\left[\frac{(v_{j_z}-v_{i_z})n_{ij_y}+(v_{i_y}-v_{j_y})n_{ij_z}}{\|\vec{\mathbf{N}}_{ij}\|}\|\vec{\mathbf{N}}_{jk}\|+\right. \\
&\left.+\frac{(v_{k_z}-v_{j_z})n_{jk_y}+(v_{j_y}-v_{k_y})n_{jk_z}}{\|\vec{\mathbf{N}}_{jk}\|}\|\vec{\mathbf{N}}_{ij}\|\right]\Bigg)\Bigg/(\|\vec{\mathbf{N}}_{ij}\|\|\vec{\mathbf{N}}_{jk}\|)^2 \\
\frac{\partial(\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk})}{\partial p_y} =& \Bigg(\big[(v_{i_z}-v_{j_z})n_{jk_x}+(v_{j_z}-v_{k_z})n_{ij_x}+ \\
&+(v_{j_x}-v_{i_x})n_{jk_z}+(v_{k_x}-v_{j_x})n_{ij_z}\big]\,\|\vec{\mathbf{N}}_{ij}\|\|\vec{\mathbf{N}}_{jk}\|- \\
&-\vec{\mathbf{N}}_{ij}\cdot\vec{\mathbf{N}}_{jk}\left[\frac{(v_{i_z}-v_{j_z})n_{ij_x}+(v_{j_x}-v_{i_x})n_{ij_z}}{\|\vec{\mathbf{N}}_{ij}\|}\|\vec{\mathbf{N}}_{jk}\|+\right. \\
&\left.+\frac{(v_{j_z}-v_{k_z})n_{jk_x}+(v_{k_x}-v_{j_x})n_{jk_z}}{\|\vec{\mathbf{N}}_{jk}\|}\|\vec{\mathbf{N}}_{ij}\|\right]\Bigg)\Bigg/(\|\vec{\mathbf{N}}_{ij}\|\|\vec{\mathbf{N}}_{jk}\|)^2 \\
\frac{\partial(\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk})}{\partial p_z} =& \Bigg(\big[(v_{j_y}-v_{i_y})n_{jk_x}+(v_{k_y}-v_{j_y})n_{ij_x}+ \\
&+(v_{i_x}-v_{j_x})n_{jk_y}+(v_{j_x}-v_{k_x})n_{ij_y}\big]\,\|\vec{\mathbf{N}}_{ij}\|\|\vec{\mathbf{N}}_{jk}\|- \\
&-\vec{\mathbf{N}}_{ij}\cdot\vec{\mathbf{N}}_{jk}\left[\frac{(v_{j_y}-v_{i_y})n_{ij_x}+(v_{i_x}-v_{j_x})n_{ij_y}}{\|\vec{\mathbf{N}}_{ij}\|}\|\vec{\mathbf{N}}_{jk}\|+\right. \\
&\left.+\frac{(v_{k_y}-v_{j_y})n_{jk_x}+(v_{j_x}-v_{k_x})n_{jk_y}}{\|\vec{\mathbf{N}}_{jk}\|}\|\vec{\mathbf{N}}_{ij}\|\right]\Bigg)\Bigg/(\|\vec{\mathbf{N}}_{ij}\|\|\vec{\mathbf{N}}_{jk}\|)^2
\end{aligned}$$

B.3.3 Derivation of the Expression $(\nabla(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk}))_{\mathbf{p}}$

The normal $\vec{\mathbf{n}}_{jk}$ is deduced from Equation (B.9) while the unit vector $\vec{\mathbf{q}}_i$ is defined as

$$\begin{aligned}
\vec{\mathbf{Q}}_i &= \big((v_{i_x}-p_x),(v_{i_y}-p_y),(v_{i_z}-p_z)\big) \\
\vec{\mathbf{q}}_i &= \frac{\vec{\mathbf{Q}}_i}{\|\vec{\mathbf{Q}}_i\|}.
\end{aligned}$$

Therefore, the scalar product $\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk}$ is as follows

$$\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk}=\frac{q_{i_x}n_{jk_x}+q_{i_y}n_{jk_y}+q_{i_z}n_{jk_z}}{\|\vec{\mathbf{q}}_i\|\|\vec{\mathbf{n}}_{jk}\|}.$$

Then, the gradient $(\nabla(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk}))_{\mathbf{p}}$, is expressed as

$$(\nabla(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk}))_{\mathbf{p}}=\left(\frac{\partial(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk})}{\partial p_x},\frac{\partial(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk})}{\partial p_y},\frac{\partial(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk})}{\partial p_z}\right),$$

where the partial derivatives are

$$\begin{aligned}\frac{\partial(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk})}{\partial p_x} =& \Bigg[-\big(n_{jk_x}+(v_{j_z}-v_{k_z})Q_{i_y}+(v_{k_y}-v_{j_y})Q_{i_z}\big)\,\|\vec{\mathbf{N}}_{jk}\|\|\vec{\mathbf{Q}}_i\|- \\ &-\vec{\mathbf{N}}_{jk}\cdot\vec{\mathbf{Q}}_i\left(\frac{(v_{k_z}-v_{j_z})n_{jk_y}+(v_{j_y}-v_{k_y})n_{jk_z}}{\|\vec{\mathbf{N}}_{jk}\|}\|\vec{\mathbf{Q}}_i\|-\right. \\ &\left.-\frac{Q_{i_x}}{\|\vec{\mathbf{Q}}_i\|}\|\vec{\mathbf{N}}_{jk}\|\right)\Bigg]\Big/(\|\vec{\mathbf{N}}_{jk}\|\|\vec{\mathbf{Q}}_i\|)^2 \\ \frac{\partial(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk})}{\partial p_y} =& \Bigg[-\big((v_{k_z}-v_{j_z})Q_{i_x}+n_{jk_y}+(v_{j_x}-v_{k_x})Q_{i_z}\big)\,\|\vec{\mathbf{N}}_{jk}\|\|\vec{\mathbf{Q}}_i\|- \\ &-\vec{\mathbf{N}}_{jk}\cdot\vec{\mathbf{Q}}_i\left(\frac{(v_{j_z}-v_{k_z})n_{jk_x}+(v_{k_x}-v_{j_x})n_{jk_z}}{\|\vec{\mathbf{N}}_{jk}\|}\|\vec{\mathbf{Q}}_i\|-\right. \\ &\left.-\frac{Q_{i_y}}{\|\vec{\mathbf{Q}}_i\|}\|\vec{\mathbf{N}}_{jk}\|\right)\Bigg]\Big/(\|\vec{\mathbf{N}}_{jk}\|\|\vec{\mathbf{Q}}_i\|)^2 \\ \frac{\partial(\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk})}{\partial p_z} =& \Bigg[-\big((v_{j_y}-v_{k_y})Q_{i_x}+(v_{k_x}-v_{j_x})Q_{i_y}+n_{jk_z}\big)\,\|\vec{\mathbf{N}}_{jk}\|\|\vec{\mathbf{Q}}_i\|- \\ &-\vec{\mathbf{N}}_{jk}\cdot\vec{\mathbf{Q}}_i\left(\frac{(v_{k_y}-v_{j_y})n_{jk_x}+(v_{j_x}-v_{k_x})n_{jk_y}}{\|\vec{\mathbf{N}}_{jk}\|}\|\vec{\mathbf{Q}}_i\|-\right. \\ &\left.-\frac{Q_{i_z}}{\|\vec{\mathbf{Q}}_i\|}\|\vec{\mathbf{N}}_{jk}\|\right)\Bigg]\Big/(\|\vec{\mathbf{N}}_{jk}\|\|\vec{\mathbf{Q}}_i\|)^2\end{aligned}$$

B.3.4 Final Derivation of the Mean Value Coordinates

In the sections above, we have presented the derivation of the individual terms of the equation that defines the Mean Value Coordinates. Now, we put them all together into the final result. However, we first define the variables $\mathcal{N}$ and $\mathcal{D}$ to rename the numerator and the denominator of the expression for $\omega_i(\mathbf{p})$, that is

$$\omega_i(\mathbf{p})=\frac{\sum_{T\ni\mathbf{v_i}}\frac{\gamma_{jk}+\gamma_{ij}\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk}+\gamma_{ki}\vec{\mathbf{n}}_{ki}\cdot\vec{\mathbf{n}}_{jk}}{2\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk}}}{d_i}=\frac{\mathcal{N}}{\mathcal{D}},$$

so,

$$(\nabla\omega_i(\mathbf{p}))_{\mathbf{p}}=\frac{(\nabla\mathcal{N})_{\mathbf{p}}\cdot\mathcal{D}+\mathcal{N}\cdot(\nabla\mathcal{D})_{\mathbf{p}}}{\mathcal{D}^2}.$$

The gradient of the numerator $\mathcal{N}$ with respect to $\mathbf{p}$ is defined as follows

$$\begin{aligned}(\nabla\mathcal{N})_{\mathbf{p}}=\sum_{T\ni\mathbf{v_i}}&\big([\,(\nabla\gamma_{jk})_{\mathbf{p}}+((\nabla\gamma_{ij})_{\mathbf{p}}\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk}+\gamma_{ij}(\nabla\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk})_{\mathbf{p}})+ \\ &+\ ((\nabla\gamma_{ki})_{\mathbf{p}}\vec{\mathbf{n}}_{ki}\cdot\vec{\mathbf{n}}_{jk}+\gamma_{ki}(\nabla\vec{\mathbf{n}}_{ki}\cdot\vec{\mathbf{n}}_{jk})_{\mathbf{p}})]\,2\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk})- \\ &-\ [\gamma_{jk}+\gamma_{ij}\vec{\mathbf{n}}_{ij}\cdot\vec{\mathbf{n}}_{jk}+\gamma_{ki}\vec{\mathbf{n}}_{ki}\cdot\vec{\mathbf{n}}_{jk}]\,2(\nabla\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk})_{\mathbf{p}})\,/\,(2\vec{\mathbf{q}}_i\cdot\vec{\mathbf{n}}_{jk})^2\,.\end{aligned}$$

Besides this, we also know that $d_i = \|\vec{\mathbf{Q}}_i\|$, so the gradient of the denominator is

$$(\nabla \mathcal{D})_{\mathbf{p}} = (\nabla d_i)_{\mathbf{p}} = \left(\frac{\partial \vec{\mathbf{Q}}_i}{p_x}, \frac{\partial \vec{\mathbf{Q}}_i}{p_y}, \frac{\partial \vec{\mathbf{Q}}_i}{p_z} \right) = \left(\frac{Q_{i_x}}{\|\vec{\mathbf{Q}}_i\|}, \frac{Q_{i_y}}{\|\vec{\mathbf{Q}}_i\|}, \frac{Q_{i_z}}{\|\vec{\mathbf{Q}}_i\|} \right).$$

Finally, the derivation of the Mean Value Coordinate $\alpha_i(\mathbf{p})$ is

$$(\nabla \alpha_i(\mathbf{p}))_{\mathbf{p}} = \frac{(\nabla \omega_i(\mathbf{p}))_{\mathbf{p}} \sum_{j=1}^{n} \omega_j(\mathbf{p}) - \omega_i(\mathbf{p}) \sum_{j=1}^{n} (\nabla \omega_j(\mathbf{p}))_{\mathbf{p}}}{\left(\sum_{j=1}^{n} \omega_j(\mathbf{p}) \right)^2}.$$

Bibliography

[AB97] F. Aubert and D. Bechmann. Volume-preserving space deformation. *Computers & Graphics*, 21(5):625–639, 1997.

[Abr74] M. Abramowitz. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,*. Dover Publications, Incorporated, 1974.

[ACOL00] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 157–164. ACM Press/Addison-Wesley Publishing Co., 2000.

[ACWK06] A. Angelidis, M. P. Cani, G. Wyvill, and S. King. Swirling-sweepers: Constant-volume modeling. *Graphical Models*, 68(4):324–332, 2006.

[Apo69] T. M. Apostol. *Calculus. 2. Multi-variable calculus and linear algebra, with applications to differential equations and probability.* Wiley, 1969.

[AS07] A. Angelidis and K. Singh. Kinodynamic skinning using volume-preserving deformations. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 129–140. Eurographics Association, 2007.

[AWC04] A. Angelidis, G. Wyvill, and M. P. Cani. Sweepers: swept user-defined tools for modeling by deformation. In *Shape Modeling International*, pages 63–73. IEEE, June 2004.

[BB91] P. Borrel and D. Bechmann. Deformation of n-dimensional objects. In *Proceedings of the 1st ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications*, SMA '91, pages 351–369. ACM, 1991.

[BBB$^+$14] A. H. Bermano, D. Bradley, T. Beeler, F. Zund, D. Nowrouzezahrai, I. Baran, O. Sorkine-Hornung, H. Pfister, R. W. Sumner, B. Bickel, and M. Gross. Facial performance enhancement using dynamic shape space analysis. *ACM Transactions on Graphics*, 33(2):13:1–13:12, 2014.

[BC01] J. D. Boissonnat and F. Cazals. Natural neighbor coordinates of points on a surface. *Computational Geometry: Theory and Applications*, 19(2-3):155–173, 2001.

[BK03] M. Botsch and L. Kobbelt. Multiresolution surface representation based on displacement volumes. *Computer Graphics Forum*, 22(3):483–492, 2003.

[BK04] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics*, 23(3):630–634, 2004.

[BL99] J. Bloomenthal and C. Lim. Skeletal methods of shape manipulation. In *Proceedings of the International Conference on Shape Modeling and Applications*, SMI '99, pages 44–47. IEEE Computer Society, 1999.

[BLB+08] B. Bickel, M. Lang, M. Botsch, M. A. Otaduy, and M. Gross. Pose-space animation and transfer of facial details. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 57–66. Eurographics Association, 2008.

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

[BP07] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics*, 26(3), 2007.

[BPGK06] M. Botsch, M. Pauly, M. Gross, and L. Kobbelt. Primo: Coupled prisms for intuitive surface modeling. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing*, SGP '06, pages 11–20. Eurographics Association, 2006.

[Bro11] L. E. J. Brouwer. Über abbildung von mannigfaltigkeiten. *Mathematische Annalen*, 71(1):97–115, 1911.

[BSPG06] M. Botsch, R. Sumner, M. Pauly, and M. Gross. Deformation transfer for detail-preserving surface editing. In *Vision, Modeling & Visualization*, pages 357–364, 2006.

[BTST12] G. Bharaj, T. Thorm&#x00e4;hlen, H. P. Seidel, and C. Theobalt. Automatically rigging multi-component characters. *Computer Graphics Forum*, 31(2pt3):755–764, 2012.

[BWKS11] M. Bokeloh, M. Wand, V. Koltun, and H. P. Seidel. Pattern-aware shape deformation using sliding dockers. *ACM Transactions on Graphics*, 30(6):123:1–123:10, 2011.

[CCD+11] M. Callieri, A. Chica, M. Dellepiane, I. Besora, M. Corsini, J. Moyés, G. Ranzuglia, R. Scopigno, and P. Brunet. Multiscale acquisition and presentation of very large artifacts: The case of portalada. *Journal on Computing and Cultural Heritage*, 3(4):14:1–14:20, 2011.

[CHHH15] X. Chen, J. Hu, H. He, and J. Hua. Spherical volume-preserving demons registration. *Computer Aided Design*, 58(0):99–104, 2015.

[Coq90] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. *SIGGRAPH Computer Graphics*, 24(4):187–196, 1990.

[Cow73] G. R. Cowper. Gaussian quadrature formulas for triangles. *International Journal for Numerical Methods in Engineering*, 7(3):405–408, 1973.

[CVB10] M. A. Cerveró, A. Vinacua, and P. Brunet. Volume-preserving deformation using generalized barycentric coordinates. In *XX Congreso Español de Informática Gráfica*, pages 57–66, 2010.

[CVB13] M. A. Cerveró, A. Vinacua, and P. Brunet. Cage-free spatial deformations. In *VISIGRAPP - International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, 2013.*, pages 111–114. SciTePress, 2013.

[dGGV08] F. de Goes, S. Goldenstein, and L. Velho. A hierarchical segmentation of articulated bodies. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1349–1356. Eurographics Association, 2008.

[DQ04] H. Du and H. Qin. Medial axis extraction and shape manipulation of solid objects using parabolic pdes. In *Proceedings of the 9th ACM Symposium on Solid Modeling and Applications*, SM '04, pages 25–35. Eurographics Association, 2004.

[Dun85] D. A. Dunavant. High degree efficient symmetrical Gaussian quadrature rules for the triangle. *International Journal for Numerical Methods in Engineering*, 21:1129–1148, 1985.

[FHK06] M. S. Floater, K. Hormann, and G. Kós. A general construction of barycentric coordinates over convex polygons. *Advances in Computational Mathematics*, 24(1–4):311–331, 2006.

[FKR05] M. S. Floater, G. Kós, and M. Reimers. Mean value coordinates in 3d. *Computer Aided Geometric Design*, 22(7):623–631, 2005.

[Flo03] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[GGPCP13] F. González-García, T. Paradinas, N. Coll, and G. Patow. *cages: A multilevel, multi-cage-based system for mesh deformation. *ACM Transactions on Graphics*, 32(3):24:1–24:13, 2013.

[HF06] K. Hormann and M. S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Transactions on Graphics*, 25(4):1424–1441, 2006.

[Hil56] F. B. Hildebrand. *Introduction to numerical analysis.* General Publishing Company, 1956.

[HML99] G. Hirota, R. Maheshwari, and M. C. Lin. Fast volume-preserving free form deformation using multi-level optimization. In *Proceedings of the 5th ACM Symposium on Solid Modeling and Applications*, SMA '99, pages 234–245. ACM, 1999.

[HRE$^+$08] C. Hecker, B. Raabe, R. W. Enslow, J. DeWeese, J. Maynard, and K. van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics*, 27(3):27:1–27:11, 2008.

[HSL$^+$06] J. Huang, X. Shi, X. Liu, K. Zhou, L. Y. Wei, S. H. Teng, H. Bao, B. Guo, and H. Y. Shum. Subspace gradient domain mesh deformation. *ACM Transactions on Graphics*, 25(3):1126–1134, 2006.

[IMH05] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Transactions on Graphics*, 24(3):1134–1141, 2005.

[JBPS11] A. Jacobson, I. Baran, J. Popović, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Transactions on Graphics*, 30(4):78:1–78:8, 2011.

[JMD$^+$07] P.r Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics*, 26(3), 2007.

[JP$^+$15] A Jacobson, D Panozzo, et al. libigl: A simple C++ geometry processing library, 2015. http://libigl.github.io/libigl/.

[JS11] A. Jacobson and O. Sorkine. Stretchable and twistable bones for skeletal shape deformation. *ACM Transactions on Graphics*, 30(6):165:1–165:8, 2011.

[JSW05] T. Ju, S. Schaefer, and J. Warren. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics*, 24(3):561–566, 2005.

[JSWD05] T. Ju, S. Schaefer, J. Warren, and M. Desbrun. A geometric construction of coordinates for convex polyhedra using polar duals. In *Proceedings of the 3rd Eurographics Symposium on Geometry Processing*, SGP '05. Eurographics Association, 2005.

[JZ07] V. Jain and H. Zhang. A spectral approach to shape-based retrieval of articulated 3d models. *Computer Aided Design*, 39:398–407, 2007.

[JZvdP$^+$08] T. Ju, Q. Y. Zhou, M. van de Panne, D. Cohen-Or, and U. Neumann. Reusable skinning templates using cage-based deformations. *ACM Transactions on Graphics*, 27(5):122:1–122:10, 2008.

[KBS00] L. Kobbelt, T. Bareuther, and H. P. Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum*, 19(3):249–260, 2000.

[KCvO08] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics*, 27(4):105:1–105:23, 2008.

[KCVS98] L. Kobbelt, S. Campagna, J. Vorsatz, and H. P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 105–114. ACM, 1998.

[KMP07] M. Kilian, N. J. Mitra, and H. Pottmann. Geometric modeling in shape space. *ACM Transactions on Graphics*, 26(3), 2007.

[KO03] K. G. Kobayashi and K. Ootsubo. t-ffd: Free-form deformation by using triangular mesh. In *Proceedings of the 8th ACM Symposium on Solid Modeling and Applications*, SM '03, pages 226–234. ACM, 2003.

[KT03] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, 2003.

[LBS06] T. Langer, A. Belyaev, and H. P. Seidel. Spherical barycentric coordinates. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing*, SGP '06, pages 81–88. Eurographics Association, 2006.

[LCF00] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th International Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00. ACM, 2000.

[LCOGL07] Y. Lipman, D. Cohen-Or, R. Gal, and D. Levin. Volume and shape preservation via moving frame manipulation. *ACM Transactions on Graphics*, 26(1), 2007.

[LG14] Z. Levi and C. Gotsman. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE Transactions on Visualization and Computer Graphics*, 21(2):264–277, 2014.

[LKA06] J. M. Lien, J. Keyser, and N. M. Amato. Simultaneous shape decomposition and skeletonization. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling*, SPM '06, pages 219–228. ACM, 2006.

[LKCOL07] Y. Lipman, J. Kopf, D. Cohen-Or, and D. Levin. Gpu-assisted positive mean value coordinates for mesh deformations. In *Proceedings of the 5th Eurographics Symposium on Geometry Processing*, SGP '07, pages 117–123. Eurographics Association, 2007.

[LL10] Y. Lipman and D. Levin. Derivation and analysis of green coordinates. *Computational Methods and Function Theory*, 10(1):167–188, 2010.

[LLCO08] Y. Lipman, D. Levin, and D. Cohen-Or. Green coordinates. *ACM Transactions on Graphics*, 27(3):78:1–78:10, 2008.

[LLD$^+$10] Z. Li, D. Levin, Z. Deng, D. Liu, and X. Luo. Cage-free local deformations using green coordinates. *The Visual Computer*, 26(6-8):1027–1036, 2010.

[LS08] T. Langer and H. P. Seidel. Higher order barycentric coordinates. *Computer Graphics Forum*, 27(2):459–466, 2008.

[LSCO$^+$04] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rössl, and H. P. Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of the Shape Modeling International 2004*, SMI '04, pages 181–190. IEEE Computer Society, 2004.

[LV99] F. Lazarus and A. Verroust. Level set diagrams of polyhedral objects. In *Proceedings of the 5th ACM Symposium on Solid Modeling and Applications*, SMA '99, pages 130–140. ACM, 1999.

[LZ07] R. Liu and H. Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum*, 26(3):385–394, 2007.

[MBK07] M. Marinov, M. Botsch, and L. Kobbelt. Gpu-based multiresolution deformation using approximate normal field reconstruction. *Journal of Graphics Tools*, 12(1):27–46, 2007.

[MBLD02] M. Meyer, A. Barr, H. Lee, and M. Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools*, 7(1):13–22, 2002.

[MERT12] J. Martinez-Esturo, C. Rössl, and H. Theisel. Continuous deformations by isometry preserving shape integration. In *Curves and Surfaces*, volume 6920 of *Lecture Notes in Computer Science*, pages 456–472. Springer Berlin Heidelberg, 2012.

[MJ96] R. MacCracken and K. I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of the 23rd International Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 181–188. ACM, 1996.

[Mon14] E. Monclús. *Advanced interaction techniques for medical models.* PhD thesis, Universitat Politècnica de Catalunya, July 2014.

[MS10] J. Manson and S. Schaefer. Moving least squares coordinates. *Computer Graphics Forum*, 29(5):1517–1524, 2010.

[MTLT88] N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings on Graphics Interface '88*, pages 26–33. Canadian Information Processing Society, 1988.

[Mö27] A. F. Möbius. *Der Barycentrische Calcul.* Johann Ambrosius Barth, 1827.

[New87] I. Newton. *Philosophiae naturalis principia mathematica.* S. Pepys, Reg. Soc. Praeses, 1687.

[NISA07] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: Designing freeform surfaces with 3d curves. *ACM Transactions on Graphics*, 26(3), 2007.

[NSACO05] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. *ACM Transactions on Graphics*, 24(3):1142–1147, 2005.

[OZS08] V. Orvalho, E. Zacur, and A. Susin. Transferring the rig and animations from a character to different face models. *Computer Graphics Forum*, 27(8):1997–2012, 2008.

[PLHY13] W. Peng, D. Lu, T. Huang, and R. Yin. As-rigid-as-possible mesh deformation and its application in hexahedral mesh generation. *Advances in Engineering Software*, 65(0):158–167, 2013.

[RHC09] D. Rohmer, S. Hahmann, and M. P. Cani. Exact volume preserving skinning with shape control. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, pages 83–92. ACM, 2009.

[RLN06] T. Rhee, J. P. Lewis, and U. Neumann. Real-time weighted pose-space deformation on the gpu. *Computer Graphics Forum*, 25(3):439–448, 2006.

[RSB95] A. Rappoport, A. Sheffer, and M. Bercovier. Volume-preserving free-form solid. In *Proceedings of the 3th ACM Symposium on Solid Modeling and Applications*, SMA '95, pages 361–372. ACM, 1995.

[SA07] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the 5th Eurographics Symposium on Geometry Processing*, SGP '07, pages 109–116. Eurographics Association, 2007.

[SCOL$^+$04] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H. P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 175–184. ACM, 2004.

[SDC09] D. Sýkora, J. Dingliana, and S. Collins. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '09, pages 25–33. ACM, 2009.

[SJW07] S. Schaefer, T. Ju, and J. Warren. A unified, integral construction for coordinates over closed curves. *Computer Aided Geometric Design*, 24(8-9):481–493, 2007.

[Sor06] O. Sorkine. Differential representations for mesh processing. *Computer Graphics Forum*, 25(4):789–807, 2006.

[SP86] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Computer Graphics*, 20(4):151–160, 1986.

[SRC01] P. P. J. Sloan, C. F. Rose, III, and M. F. Cohen. Shape by example. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, pages 135–143. ACM, 2001.

[Sug02] K. Sugihara. Laguerre voronoi diagram on the sphere. *Journal for Geometry and Graphics*, 6(1):69–81, 2002.

[vFTS06] W. von Funck, H. Theisel, and H. P. Seidel. Vector field based shape deformations. *ACM Transactions on Graphics*, 25(3):1118–1125, 2006.

[vFTS07] W. von Funck, H. Theisel, and H. P. Seidel. Explicit control of vector field based shape deformations. In *Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on*, pages 291–300, 2007.

[vFTS08] W. von Funck, H. Theisel, and H. P. Seidel. Volume-preserving mesh skinning. In *Proceedings of Vision, Modeling, and Visualization (VMV 2009)*, pages 407–414. Akademische Verlagsgesellschaft AKA, 2008.

[VMHB14] L. Váša, S. Marras, K. Hormann, and G. Brunnett. Compressing dynamic meshes with geometric laplacians. *Computer Graphics Forum*, 33(2):145–154, 2014.

[VNB10] A. Vinacua, I. Navazo, and P. Brunet. Baip 2020, buque autónomo inteligente pesquero 2020, 2008–2010.

[Wac75] E. L. Wachspress. *A Rational Finite Element Basis*. Academic Press, 1975.

[WBCG09] O. Weber, M. Ben-Chen, and C. Gotsman. Complex barycentric coordinates with applications to planar shape deformation. *Computer Graphics Forum*, 28(2):587–597, 2009.

[WM14] T. Wu and K. Mithraratne. A volume-preserving free-form deformation technique for customising a face model to another configuration. In *The 15th International Conference on Biomedical Engineering*, volume 43 of *IFMBE Proceedings*, pages 645–648. Springer International Publishing, 2014.

[WML$^+$03] F. Wu, W. Ma, P. Liou, R. H. Laing, and M. Ouhyoung. Skeleton extraction of 3d objects with visible repulsive force. In *Proceedings of Computer Graphics Workshop*, 2003.

[WSHD07] J. Warren, S. Schaefer, A. N. Hirani, and M. Desbrun. Barycentric coordinates for convex sets. *Advances in Computational Mathematics*, 27(3):319–338, 2007.

[WSLG07] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. Context-aware skeletal shape deformation. *Computer Graphics Forum*, 26(3):265–273, 2007.

[YBS03] S. Yoshizawa, A. G. Belyaev, and H. P. Seidel. Free-form skeleton-driven mesh deformations. In *Proceedings of the 8th ACM Symposium on Solid Modeling and Applications*, SM '03, pages 247–253. ACM, 2003.

[YBS07] S. Yoshizawa, A. G. Belyaev, and H. P. Seidel. Skeleton-based variational mesh deformations. *Computer Graphics Forum*, 26(3):255–264, 2007.

[YHM06] H. B. Yan, S. M. Hu, and R. Martin. Skeleton-based shape deformation using simplex transformations. In *Proceedings of the 24th International Conference on Advances in Computer Graphics*, CGI'06, pages 66–77. Springer-Verlag, 2006.

[YZX$^+$04] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H. Y. Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics*, 23(3):644–651, 2004.

[ZDL$^+$14] J. Zhang, B. Deng, Z. Liu, G. Patanè, S. Bouaziz, K. Hormann, and L. Liu. Local barycentric coordinates. *ACM Transactions on Graphics*, 33(6):188:1–188:12, 2014.

[ZHM11] S. Zhang, J. Huang, and D. N. Metaxas. Robust mesh editing using laplacian coordinates. *Graphical Models*, 73(1):10–19, 2011.

[ZHS$^+$05] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H. Y. Shum. Large mesh deformation using the volumetric graph laplacian. *ACM Transactions on Graphics*, 24(3):496–503, 2005.

[ZRKS05] R. Zayer, C. Rössl, Z. Karni, and H. P. Seidel. Harmonic guidance for surface deformation. *Computer Graphics Forum*, 24:601–609, 2005.

[ZSGS04] K. Zhou, J. Synder, B. Guo, and H. Y. Shum. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 45–54. ACM, 2004.

[ZT99] Y. Zhou and A. W. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, 1999.