



HAL
open science

3D mesh morphing

Bogdan Cosmin Mocanu

► **To cite this version:**

Bogdan Cosmin Mocanu. 3D mesh morphing. Economics and Finance. Institut National des Télécommunications, 2012. English. NNT : 2012TELE0049 . tel-00836048

HAL Id: tel-00836048

<https://theses.hal.science/tel-00836048>

Submitted on 20 Jun 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



THESE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité : Informatique et Télécommunications

Ecole doctorale : Informatique, Télécommunications et Electronique de Paris

Présentée par

Bogdan Cosmin MOCANU

**Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS**

Métamorphose de maillages 3D

Soutenue le 29 Novembre 2012

devant le jury composé de :

Président de jury :	Madame le Maître de Conférences, HDR	Catherine ACHARD
Rapporteur :	Monsieur le Professeur	Danco DAVCEV
Rapporteur :	Monsieur le Professeur	Malik MALLEM
Examineur :	Monsieur le Professeur	Teodor PETRESCU
Directeur de thèse :	Monsieur le Professeur	Titus ZAHARIA

These N° : 2012TELE0049

3D Mesh Morphing

Acknowledgement

First and foremost, I am especially grateful to my thesis supervisor, Professor Titus Zaharia, for his guidance throughout my PhD thesis, and especially for giving me the freedom to pursue my own ideas in the area of computer vision (video indexing). Professor Titus Zaharia offered me invaluable help, starting from his academic advice and very helpful hints in the research proposed in this thesis.

Special thanks I would like to address to the members of my Ph.D. defense committee.

First, I would like to express all my gratitude to Professors Danco Davcev from University for Information Science and Technology and Malik Mallem, from Université d'Evry Val d'Essonne for accepting the task of being reviewers. Their reviews, comments and fruitful suggestions helped me to improve the manuscript and to give its final shape.

To Professor Catherine Achard, from Université Pierre et Marie Curie – Paris VI, I would like to express all my thanks for her interest in my research work.

My warm thanks are also going to Professor Teodor Petrescu, from the University Politehnica of Bucharest, for his support and advices during all these years.

Being a part of the ARTEMIS Department within Institut Mines-Télécom/Télécom SudParis was for me a great experience. I have worked closely with many members of the department. In particular, I would like to thank here Raluca, Adriana, Afef and Andrei for providing me with such an enjoyable and stimulating working environment. Also, I would like to thank Marius Preda, Mihai Mitrea and Cătălin Fetița, Associate Professors within the ARTEMIS department, for their attentive listening and stimulating help.

To Madame Evelyne Taroni I would like to express my gratitude for her help, patience and inexhaustible energy in solving all administrative problems. Also, I would like to thank Madame Marilyn Galopin from the Doctoral School EDITE de Paris, for her invaluable help in administrative issues.

I would finally like to thank my family and friends, both near and far, for providing me endless support, motivation, and inspiration. The unconditional love of my parents and sister was an unwavering source of strength throughout the writing of this thesis. Last, but not least, I would like to acknowledge the never-ending patience and support of my fiancée, Ruxandra. Without her support, none of my achievements would have been possible.

Table of Contents

I. INTRODUCTION	1
II. 3D MODEL REPRESENTATIONS	5
II.1. THE 3D VIRTUAL WORLD	6
II.2. 3D MESH REPRESENTATIONS	8
II.2.1. Definitions and terminology	10
II.2.2. File formats for mesh storage.....	14
II.3. CONCLUSIONS.....	16
III. AN OVERVIEW OF 3D MESH MORPHING TECHNIQUES.....	17
III.1. MORPHING IN GENERAL.....	18
III.2. VISUALLY PLAUSIBLE MORPHING	19
III.3. 2D IMAGE MORPHING	20
III.4. 3D OBJECT MORPHING	25
III.4.1. 3D volume based morphing	25
III.4.2. 3D mesh morphing	29
III.5. OVERVIEW OF THE PROPOSED 3D MESH MORPHING FRAMEWORK	48
III.6. CONCLUSIONS.....	49
IV. MESH PARAMETERIZATION	51
IV.1. INTRODUCTION	52
IV.2. PLANAR PARAMETERIZATION OF TRIANGULAR MESHES.....	55
IV.2.1. Selection of the boundary's shape	58
IV.2.2. Spring weights specification	60
IV.2.3. Edge length ratio preserving (ELRP) planar parameterization	73
IV.2.4. Objective experimental evaluation of planar mesh parameterization methods ...	75
IV.3. SPHERICAL PARAMETERIZATION OF TRIANGULAR MESHES	79
IV.3.1. State of the art on spherical embedding	79
IV.3.2. Curvature-driven spherical parameterization	89

IV.3.3. Experimental evaluation	94
IV.4. CONCLUSIONS	98
V. MESH DEFORMATION FOR FEATURE ALIGNMENT	101
V.1. INTRODUCTION	102
V.2. RELATED WORK	103
V.2.1. Space deformations.....	103
V.2.2. Free-form deformations	104
V.2.3. Skeletal deformation	106
V.2.4. Multiresolution mesh editing	107
V.2.5. Laplacian mesh editing	108
V.2.6. Radial basis functions.....	112
V.3. MESH QUALITY METRICS	115
V.4. EXPERIMENTAL EVALUATION.....	117
V.4.1. Deformation in 2D test cases.....	118
V.4.2. Deformation in 3D test cases.....	125
V.5. FEATURE ALIGNMENT BASED ON MESH WARPING	131
V.6. CONCLUSIONS.....	132
VI. SUPERMESH CONSTRUCTION AND INTERPOLATION.....	133
VI.1. INTRODUCTION	134
VI.2. TOPOLOGY MERGING FOR MESH MORPHING.....	134
VI.3. ADAPTIVE PSEUDO-METAMESH CONSTRUCTION	144
VI.4. MESH INTERPOLATION.....	150
VI.5. CONCLUSIONS	153
VII. CONCLUSIONS AND FUTURE WORK.....	154
List of publications.....	157
REFERENCES.....	159

List of Figures

Figure II.1. 3D Model.....	6
Figure II.2. Cartesian coordinate system: (a) with the x-axis pointing toward the viewer, (b) with the z axis pointing toward the viewer (used in computer graphics).....	7
Figure II.3. World space, model space and camera space	7
Figure II.4. 3D model represented as a triangular mesh.	9
Figure II.5. Face - vertex meshes representation.....	10
Figure II.6. 3D models with various topologies.	11
Figure II.7. Two manifold meshes: (a) closed fan; (b) open fan.	12
Figure II.8. Non manifold meshes.....	12
Figure II.9. The concept of orientable mesh: (a) orientable mesh; (b) non orientable mesh. .	12
Figure III.1. Two morphing sequences between a fish and a duck model [Ath10].	18
Figure III.2. Volume representation of a stack of images: (a) without, (b) with interpolated images [Rup94a].	21
Figure III.3. Image Morphing: (a) Cross-dissolve; (b) Warping and cross dissolve [Gom99].	22
Figure III.4. Different types of warps: (a) original image; (b) shift to the right; (c) scaling in the vertical direction; (d) shear; (e) scaling in the horizontal direction; (f) rotation; (g) quadratic.	22
Figure III.5. Image mesh warping [Wol98].	23
Figure III.6. 3D cross dissolve based morphing [Efr11].	26
Figure III.7. Wavelet Domain Volume Morphing[He94].	26
Figure III.8. A volumetric field morphing sequence: (a) source volume; (b)-(h) in-between volumes; (i) target volume; (j) disk fields that control the sphere-to-head morphing[Che99].	27
Figure III.9. Morphing by mesh warping[Efr11].	28
Figure III.10. Examples of 3D mesh parameterizations.....	29
Figure III.11. Overlaid parameterization of two spherical mappings.	30
Figure III.12. Morphing between the models of a young pig and a grown-up pig. In the upper row, no feature is specified, which leads to unpleasant effects (8 legs). In the lower row, the eyes, ears, legs, and the tail are put into correspondence yielding a more natural transformation [Ale02].	31
Figure III.13. Generic 3D mesh morphing scheme.....	32
Figure III.14. Maps overlapping and boundary control vertex alignment.	33
Figure III.15. Supermesh construction.	33
Figure III.16. (a) Source model; (b) the base domain of the source model; (c) target model; (d) the base domain of the target model; [Lee99]	35

Figure III.17. Morph sequence depending on location [Lee99].....	36
Figure III.18. MIMesh algorithm: (1) Base interpolation mesh construction; (2) Mesh interpolation and parameterization; (3) Subdivision fitting scheme [Mic01].	37
Figure III.19. Normal Map [Mic01].....	37
Figure III.20. MIMesh approximation: (a) input model; (b) interpolated mesh without normal maps; (c) interpolated mesh using normal maps [Mic01].	37
Figure III.21. Local morphing: (a) source model; (b) locally interpolated mesh; (c) target model [Ale02].	38
Figure III.22. Morphing of scheduled interpolation in wavelet domain [Yu03].	39
Figure III.23. Multi-target morphing [Yu03].....	39
Figure III.24. (a) Pyramid coordinates; (b) tangential components in the projection plane Π ; (c) normal component β	41
Figure III.25. Primitive operations used to transform the mesh connectivity: (a) ESO – edge swap operation; (b) VRO – vertex removal operation; VSO – vertex split operation.	43
Figure III.26. The process of connectivity transformation employed in [Cha05].	44
Figure III.27. The process of metamesh construction [Ath12].	47
Figure III.28. Steps involved in our morphing process.	48
Figure IV.1. Vertex to vertex correspondences.	52
Figure IV.2. Mesh parameterization.....	53
Figure IV.3. Fold-overs which lead to an invalid parameterization: (a) Boundary intersection; (b) Triangle flip.	54
Figure IV.4. Isometric parameterization of a cylinder.	55
Figure IV.5. Parameterization of a spring model: (a) original spring system; (b) parameterization with fixed boundary; (c) system relaxation.	56
Figure IV.6. Parameterization with different bounding polygons[Lee02]: (a) 3D original mesh; (b) circle; (c) square; (d) free boundary.	59
Figure IV.7. Angles used for weights computation.	61
Figure IV.8. Edge and angle notation used in [Hor00].	62
Figure IV.9. Angles used in the case of the circle patterns method.....	63
Figure IV.10. Incompatibility of edge length in a wheel paradigm [She01].	65
Figure IV.11. Equiareal mapping [Flo05]: In the three cases, the areas of the corresponding cells are identical.....	67
Figure IV.12. p_i vertex projection into the triangle p_j, p_k, p_l	70
Figure IV.13. Local straightest path.	71
Figure IV.14. Parameterizing the boundary over the unit circle.....	73
Figure IV.15. The one-ring neighbors of vertex p_i and the associated lengths.	74
Figure IV.16. Comparative visual evaluation of 3D mesh planar parameterization (1).	77
Figure IV.17. Comparative visual evaluation of 3D mesh planar parameterization (2).	78

Figure IV.18. Planar parameterization of a closed genus-0 3D mesh by cuts [She02].	79
Figure IV 19. Shape projection on a circle. (a) Kent method applied on a convex shape; (b) Kent method applied on a non convex shape; (c) kernel approach.	80
Figure IV.20. Problems encountered in sphere parameterization: (a) Collapsed mapping; (b) Overlapped triangles in sphere parameterization [Ale00].	81
Figure IV.21. Spherical parameterization using relaxation approach proposed by Alexa[Ale00].	81
Figure IV.22. Spherical parameterization using the Haker's approach [Hak00].	82
Figure IV.23. Curvilinear Spherical Parameterization [Zay06]. (a) mesh cut along the date line; (b) the initial parameterization in curvilinear coordinates (with high distortions); (c) the improved mapping taking into account spherical distortion; (d) the final spherical parameterization.	83
Figure IV.24. Two textured meshes after a spherical parameterization with Birkholz approach [Bir04].	86
Figure IV.25. Planar parameterization of meshes with arbitrary genus.	87
Figure IV 26. The curvature in a point for a: (a) curve; (b) 3D surface.	90
Figure IV.27. Iterative curvature-driven flattening.	92
Figure IV.28. Vertex split operation. (a) inner vertex; (b) border vertex.	93
Figure IV.29. Vertex insertion operation: (a) initial configuration; (b) polygon subdivision; (c) set of valid positions; (d) final retained position and the new configuration.	94
Figure IV.30. Visual evaluation of our 3D mesh spherical parameterization (1).	96
Figure IV.31. Visual evaluation of our 3D mesh spherical parameterization (2).	97
Figure V.1. 3D Mesh models and associated feature points.	102
Figure V.2. Illustration of two parameterizations where feature are not aligned: (a), (d) original models; and (b), (c) their embeddings.	103
Figure V.3. Space deformations [Bar84]: (a) rotation in z – twist; (b) scale – taper; (c) rotation in y – bend.	104
Figure V.4. Free form deformation.	105
Figure V.5. Wires: A geometric deformation technique [Sin98].	105
Figure V.6. Skeleton based deformation [Yan08b].	106
Figure V.7. An example of a triangular mesh and its associated symmetric Laplacian matrix.	109
Figure V.8. Test mesh and the control rectangle with the (a) initial position and ((b), (c), (d)) final positions of rectangle corresponding to the three test cases considered.	118
Figure V.9. Quality of the worst triangle of the mesh for (a) CTPS C_a^2 and (b) CP C^2 (Case 1).	119
Figure V.10. The mean quality of all triangles in the mesh for (a) CTPS C_a^2 and (b) CP C^2 (Case 1)	119

Figure V.11. Accuracy displacement of the control points for (a) CTPS C^2_a and (b) CP C^2 (Case 1).	119
Figure V.12. CPU computational time for (a) CTPS C^2_a and (b) CP C^2 (Case 1).....	120
Figure V.13. Quality of the worst triangle of the mesh for (a) CTPS C^2_a and (b) CP C^2 (Case 2).	120
Figure V.14. The mean quality of all triangles in the mesh for (a) CTPS C^2_a and (b) CP C^2 (Case 2).	120
Figure V.15. Accuracy displacement of the control points for (a) CTPS C^2_a and (b) CP C^2 (Case 2).	120
Figure V.16. CPU computational time for (a) CTPS C^2_a and (b) CP C^2 (Case 2).....	120
Figure V.17. Quality of the worst triangle of the mesh for (a) CTPS C^2_a and (b) CP C^2 (Case 3).	121
Figure V.18. The mean quality of all triangles in the mesh for (a) CTPS C^2_a and (b) CP C^2 (Case 3).	121
Figure V.19. Accuracy displacement of the control points for (a) CTPS C^2_a and (b) CP C^2 (Case 3).	121
Figure V.20. CPU computational time for (a) CTPS C^2_a and (b) CP C^2 (Case 3).....	121
Figure V.21. The influence of the number of steps on the deformed mesh.	122
Figure V.22. RBF mesh deformation on steps.	123
Figure V.23. Visual analysis of mesh deformation in 2D space.....	124
Figure V.24. The impact of the support radius r and the number of intermediate steps over the mesh deformation for CTPS C^2_a function (Test case 3).	128
Figure V.25. Visual analysis of mesh deformation in 3D case for the RBF functions: CTPS C^2_a , CP C^2 and Gaussian.	129
Figure V.26. Visual analysis of mesh deformation in 3D case for Classical Laplacian coordinates, UFLC and TLC methods.	130
Figure V.27. Feature vertices correspondence through spherical embeddings warping (Hipo-Cow case).....	131
Figure V.28. Feature vertices correspondence through spherical embeddings warping (Igea-ManHead case).....	132
Figure VI.1. Constructing the supermesh: (a) embedding of the two connectivities in the common parametric domain; (b) Edge to edge intersection; (c) Triangulation.	135
Figure VI.2. Edge intersection algorithm of [Ken92]......	135
Figure VI.3. Edge intersection scheme labeled according to the SMCC algorithm [Lee03].	138
Figure VI.4. Different cases of intersection.	140
Figure VI.5. The three kinds of SMCC for $pstartHT$ on HS : (a) first ring; (b) 4-sided polygon; (c) a triangle.....	141
Figure VI.6. First ring neighbors retriangulation.	141

Figure VI.7. Quad-tree structure of MIMesh [Mic01].	142
Figure VI.8. Adaptive subdivision scheme to resolve the T-vertices [Mic01].	142
Figure VI.9. Vertices embedding: (a) original configuration of target vertices mapped onto a source triangle; (b) result of simple embedding; (c) enhanced result after edge swaps [Ahn04].....	143
Figure VI.10. Pop-up effect due to edge swap: (a) original mesh; (b) swapped edge	144
Figure VI.11. Point inside triangle test.	145
Figure VI.12. 1-to-4 subdivision scheme.....	147
Figure VI.13. Mesh retriangulation: (a) the pseudo-metamesh before the subdivision; (b) the pseudo-metamesh obtained after the 1-to-4 subdivision scheme; (c) retriangulated pseudo-metamesh.....	148
Figure VI.14. Pseudo metameshes: (a) original models; (b) spherical parameterization; (c) overlaid maps; (d) final pseudo metamesh.	149
Figure VI.15. Morphing between Igea and Head1 models.	150
Figure VI.16. Morphing between Cow and TRex models.	150
Figure VI.17. Morphing between Horse and Lion models.	150
Figure VI.18. Morphing between Dino and Horse models.....	151
Figure VI.19. Morphing between Armadillo and Man models.....	151
Figure VI.20. Morphing between DinoSkeleton and Dino models.	151
Figure VI.21. Graphical user interface: (a) view with the input models; (b) view during mesh simplification; (c) view during parameterization; (d) view with the final spherical embeddings.....	152

List of Tables

Table IV.1. Description of planar parameterization methods.....	72
Table IV.2. Comparative study concerning area, angles and length distortions.	76
Table IV.3. Comparison of spherical parameterization methods – Part 1.....	88
Table IV.4. Comparison of spherical parameterization methods – Part 2.....	89
Table IV.5. Comparative study concerning area and angle distortions.....	95
Table V.1. Radial basis functions.	114
Table V.2. Mesh deformation quality analysis for 2D test cases.	125
Table V.3. Mesh deformation quality analysis for 3D test cases.	127
Table VI.1. Pseudo metamesh characteristics.....	149

I. INTRODUCTION

When investigating nature we can observe that living creatures are able to change their shape over time in a smooth and gradual manner. Plants or animals are growing gradually. The growth process is a highly complex mechanism that generates internal forces which constrain organisms to modify their shape and appearance. Starting from a simple seed, a plant can grow in a complete tree, with stem, branches and leaves. Such evolutions and changes that occur in the natural world have attracted the attention of a significant number of computer science researchers who have tried over time to simulate such phenomena by computer, creating different animation techniques for shape transformation of artificial objects. Such techniques are called morphing or metamorphosis. The word metamorphosis has its origins in the Greek metamorphoum (meta – involving changes and morphoum – form), the common meaning of the word being - “a change in form or nature”.

Morphing methods are today extensively used in computer graphics to simulate the transformation between two completely different objects or to create new shapes by a combination of other existing shapes. It has a variety of applications ranging from special effects in film industry and other visual arts to medical imaging and scientific purposes.

The problem of constructing a smooth transition between two objects has been first addressed in the 2D case [Sho03], [Rah07]. Image morphing or 2D morphing consists in the construction of an image sequence representing the gradual transition between a source and a target image. Such techniques can be applied either to whole image or to some specific objects corresponding to regions of interest.

As prominent application domains that take advantage of such morphing methods, let us mention those related to cinema/television industry and notably the creation of special effects. Probably, the most popular example is the well known “Black or White” video clip produced by Michal Jackson in the early 1990’s, where such techniques where specifically applied on 2D images of human faces.

However, 2D morphing techniques present some limitations. Most often, 2D images/objects represent projections of 3D scenes/objects. As a consequence, the intermediate stages of

the 2D morphing process may not correspond to the morphing of a real 3D scene, which might lead to visually poor results.

Moreover, 3D object representations permit to generate animation sequences which are independent of the point of view. The user has in this case the flexibility to control the camera position, such that the morphing sequence can be observed from arbitrary points of view.

Elaborating advanced and efficient 3D morphing methods can have a strong economical impact on the graphics industry, specifically within the framework of content/special effects production.

This thesis specifically deals with the issue of metamorphosis of 3D objects represented as 3D meshes. The objective is to elaborate a complete 3D mesh morphing methodology able to ensure high quality transition sequences, which should be as smooth and gradual as possible, consistent with respect to both geometry and topology, and visually pleasant. From a methodological point of view, the main difficulty that has to be addressed and solved relates to the topological aspects. Thus, existing 2D image morphing methods cannot be extended in a straightforward manner to 3D meshes. 2D images are defined on a fixed and regular topology, represented as rectangular lattice of pixels. Extending such methods to 3D meshes of arbitrary and most often highly irregular connectivity, is still a challenge.

The remainder of this manuscript is organized as follows. Chapter II sets the context for morphing animation and recalls some relevant concepts related to 3D virtual environments and representations. We introduce here the background definitions and terminologies used in computer graphics and related to our work.

Chapter III presents the state-of-the-art in the field of morphing algorithms. We start analyzing the techniques proposed in the 2D morphing fields, since in this case methods have reached a mature stage and are currently extensively used in commercial applications. Next, the most representative and recent methods of 3D morphing are described and analyzed in terms of advantages and limitations.

Chapter IV first provides an overview of mesh parameterization techniques. Mesh parameterization represents a phase of utmost importance in any 3D mesh morphing approach. After reviewing the state of the art, we introduce here two different approaches which are able to construct valid parameterizations either for models with a disk topology or for closed objects topologically equivalent with a sphere. Our first approach represents an enhanced 3D object planar parameterization method introducing a new barycentric mapping algorithm based on the length ratio preservation. The second proposed approach represents

a spherical parameterization method which exploits the Gaussian curvature associated to the mesh vertices.

Chapter V covers the issue of feature alignment between the two models considered in the morphing process. The problem is solved in the parametric domain with the help of mesh warping techniques. However, not all existing deformation techniques are well suited for our purpose. Thus, we propose first an evaluation of the main warping algorithms encountered in the literature and we retain the one that meet the constraints related to feature alignment of meshes defined in the parametric domain and which lead to a minimum mesh distortion.

In Chapter VI we introduce a new method which build a pseudo metamesh that starts with the target mesh structure and is adaptively refined such that to better approximate both source and target models. Our approach avoids tracking the edge intersections between the mesh mappings of the two models and reduces drastically the number of vertices normally needed in a supermesh structure. Finally, the obtained pseudo metamesh is exploited for morphing purposes, with the help of a linear interpolation technique. Several examples of morphing between 3D objects with different characteristics are provided. Chapter VI provides also a representation of the integrated morphing system, that allows the user to control and visualize all the stages of the morphing process described in this thesis.

Finally, Chapter VII concludes our work, summarizing the main contributions proposed, and opening some perspectives of future work.

II. 3D MODEL REPRESENTATIONS

Summary: *This chapter briefly recalls background definitions and terminologies related to the 3D virtual environments, together with some popular methods exploited for model representation and different file formats used to store such complex data.*

II.1. THE 3D VIRTUAL WORLD

A 3D model is the abstract representation of an object, including structures, attributes, variation laws and relationships among components. A 3D model represents a 3D object using a collection of points in the 3D space, connected by various geometric entities such as triangles, lines, curved surfaces...

An example is presented in Figure II.1. Being a collection of data (points and other information), 3D models can be created by hand, algorithmically (procedural modeling), or scanned.

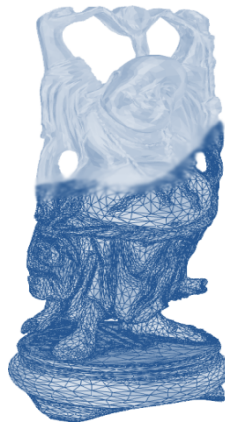


Figure II.1. 3D Model.

Bringing the objects from the real world into a virtual reality environment has always been an interesting task in a wide variety of fields. Nowadays 3D model representations are used for specific activities in different areas like:

- in the medical industries in order to construct to detailed models of organs;
- in the movie industry, where objects are animated in such a manner to simulate the real world;
- in the video gaming industries, where 3D models are used as assets for computer games;
- in the scientific sector, for various simulation purposes;
- in the architecture industries, where they are needed to illustrate proposed buildings and landscapes;
- in CAD, in order to constructs new devices, vehicles and structures based on predefined models.

Until recent years, the quality of 3D models was limited by the hardware and software capabilities. Today, the general public can easily visualize and manipulate complex models. Moreover, modern scanning technologies make it possible to generate accurate 3D models of real-life objects.

The virtual environments enable the user's interaction with the models. He can rotate, scale, deform, edit and observe the models under different lighting conditions, change their appearance (color, material, etc.), and observe the interaction with another models in the environment. Also, 3D modelers can export their models to files, which can be afterwards imported into other applications.

In all cases, a 3D Cartesian coordinate system (Oxyz) is needed in order to specify the location of the objects in space (Figure II.2).

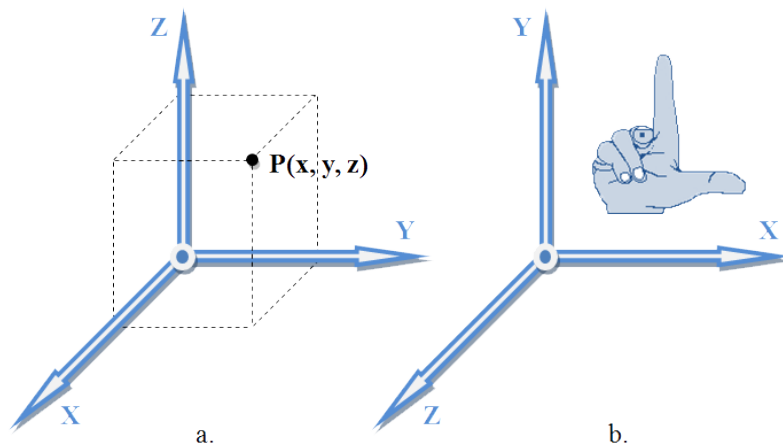


Figure II.2. Cartesian coordinate system: (a) with the x-axis pointing toward the viewer, (b) with the z axis pointing toward the viewer (used in computer graphics)

It is important to note that the coordinates axes used in computer graphics do form a right handed coordinate system. In particular, this means that the right-hand rule applies to cross products of vectors in \mathbb{R}^3 .

In practice, different coordinate systems are used. Most often, the following three common coordinate spaces are encountered in the computer graphics field (Figure II.3):

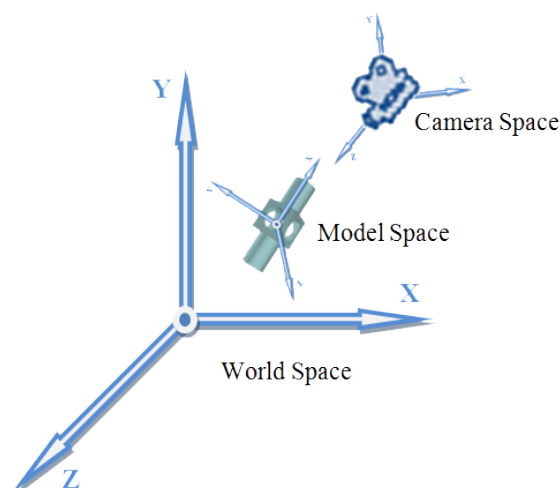


Figure II.3. World space, model space and camera space

- *Local space* (or *model space*) – is the coordinate system in which an object is defined without regard to its position, size or orientation in relation to other objects in the world. Once

the 3D model defined in the local space, it can be inserted in a global scene, by specifying the origin and axes of the local space coordinate system relative to the global scene.

- The *world coordinate system* (or *world space*) defines the locations of all geometric objects as they exist at rendering time, with all applicable transforms acting on them. The world coordinate system can be seen as a global reference system for all others coordinate systems.
- The *camera space* is a coordinate system defined relative to a virtual camera or eye that is located in world space.

Whatever the considered space, the 3D objects can be represented in various manners. In particular, we distinguish the following two main families of modeling approaches:

- Boundary representations – an object is represented by a set of surfaces (named also faces) that separate its interior from the rest of the environment. These faces are regions or subsets of closed and orientable surfaces. Each face is bounded by edges and each edge is delimited by two vertices. A boundary representation is essentially a local representation connecting faces, edges and vertices.
- Solid representations – This type of representation gives a complete and unambiguous definition of an object, describing not only the shape of the boundaries but also the object's interior and exterior regions.

In our work, we have considered solely boundary representations, and notably 3D mesh models, which are recalled in the following section.

II.2. 3D MESH REPRESENTATIONS

Polygonal 3D meshes have become the most popular object representation technique with a long history in computer vision. This increase in popularity is due to several factors including advances in computer storage capacity and processing power and the development of dense range sensors, which produce rectangular arrays of 3D points that can easily be transformed into meshes. Meshes can faithfully approximate complex free-form objects up to any desired accuracy, given sufficient space to store the representation.

A polygon mesh is a collection of vertices (points in 3D space), edges and faces that defines the shape of a polyhedral object in 3D computer graphics. The faces usually consist of triangles, quadrilaterals or other simple convex polygons, because this simplifies the rendering process, but may also be composed of more general concave polygons. In this

work, we deal only with triangular meshes since this is the most widespread mesh representation. An example of a triangular surface model is illustrated in Figure II.4.

A valid mesh structure does not contain any isolated vertices or edges. In other words, all vertices and edges are parts of triangles. Let us recall the mesh definition according to [Gar99]. A polygonal surface model $M(V, E, F)$ is a triplet containing a set of vertices V , a set of edges E and a set of triangles F . The vertex list $V = V(v_1, v_2, \dots, v_{N_V})$ includes a number of N_V elements in the form of a column vector which represent every vertex $p_i = [x_i, y_i, z_i]^T$. An edge $e_l = e_l(p_i, p_j)$, from the edge list $E = E(e_1, e_2, \dots, e_{N_E})$ of N_E elements, is defined by the two end points p_i and p_j . The face list $F = F(f_1, f_2, \dots, f_{N_F})$ contains N_F triangles, each one defined as an ordered list of three vertices identifying the corners, $f_m = f_m(p_i, p_j, p_k)$.

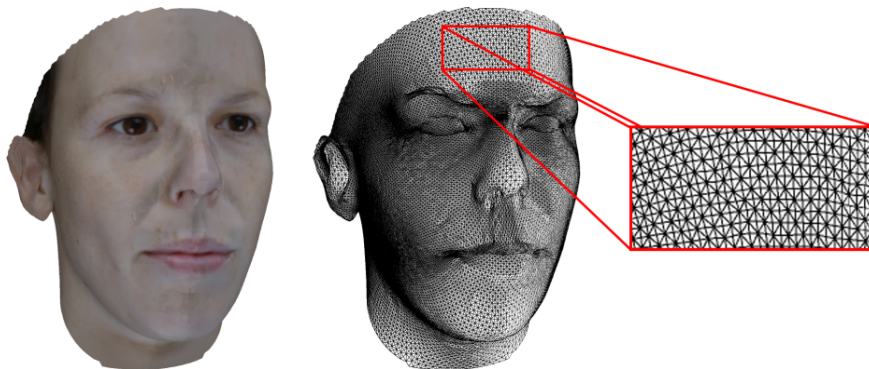


Figure II.4. 3D model represented as a triangular mesh.

The above definition of a polygonal model corresponds to a form of simplicial complex. In our case, a simplex σ is either a vertex (or 0-simplex), a line segment (1-simplex), or a triangle (2-simplex). In general, a k -simplex σ^k is the smallest closed convex set (*i.e.*, the convex hull) defined by $k + 1$ linearly independent points $\sigma^k = a_0 a_1 \dots a_k$ which are called its vertices. We can express any point p within this set as a convex combination of the vertices $p = \sum_i t_i a_i$ where $\sum_i t_i = 1$ and $t_i \in [0,1]$. Any simplex defined by a subset of the points $a_0 a_1 \dots a_k$ is a subsimplex of the simplex σ_k . A 2D simplicial complex K is a collection of vertices, edges and triangles satisfying the conditions:

- If $\sigma_i, \sigma_j \in K$, then they are either disjoint or intersect only at a common subsimplex. Specifically, two edges can only intersect at a common vertex and two faces can only intersect at a shared edge or vertex.

- If $\sigma_i \in K$, then all of its subsimplices are in K . For instance, if a triangle f is in K , then its vertices and edges must also be in K .

In practice, a 3D mesh can be completely defined only by the list of vertices and the list of triangles. The third set, the list of edges, can be obtained implicitly from the first two sets. This method to describe a 3D model is called face-vertex mesh representation. An example of a mesh stored in the face-vertex form is presented in Figure II.5.

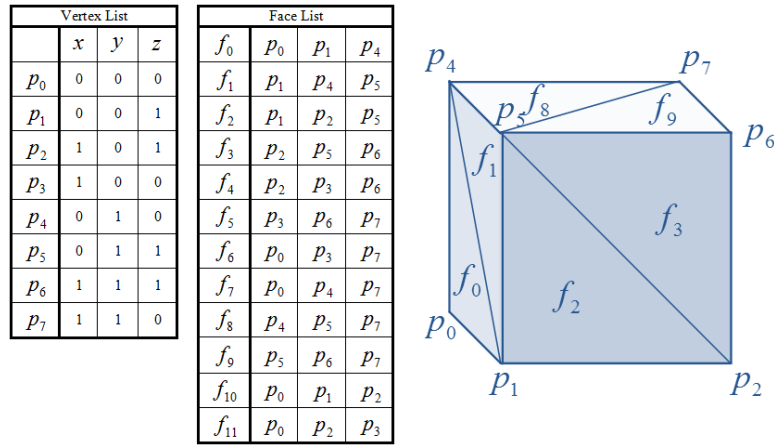


Figure II.5. Face - vertex meshes representation.

II.2.1. Definitions and terminology

In this section we review some of the most commonly terms used and in the 3D morphing field. Let us first to mention that in the following chapters only triangular 3D meshes will be considered. This is without loss of generality, since any polygonal mesh can be converted into a triangular one with the help of a triangulation method ([Bai10], [Sun09], [Nin09]).

II.2.1.1. Geometry and topology

Positions in the Euclidian space of all vertices $p_i(x_i, y_i, z_i)$ denote the *mesh geometry*, and the way how such vertices are inter-connected by edges and faces indicate its *topology*.

Actually, the term topology has two meanings, which can be distinguished by the context in which they are used. The first meaning, from traditional mathematics, refers to the local neighborhood properties of the surface of an object. This can lead to global topological characteristics, such as numbers of holes and connected components. For morphing purposes, it is important to divide the considered 3D surfaces into classes of topologically equivalent objects since within a given class, two different objects can be homeomorphically mapped into each other. (*cf.* Section II.2.1.2)

The second meaning of the term topology, popular in the computer graphics literature, refers to the vertex/edge/face connectivity of an object. Objects that are equivalent in this form of topology are identical except for the x -, y -, z -positions of their vertices (*i.e.*, the geometry of the object).

II.2.1.2. Homeomorphism

A homeomorphism can be defined as a bijective mapping $h: A \rightarrow B$ between the surfaces of two models A and B that is continuous and with the inverse function also continuous. If such

a mapping exists, we say that models A and B are homeomorphic, or topologically equivalent.

In Figure II.6 a number of models with various topologies are presented. Only the meshes from Figure II.6.a and Figure II.6.e are topologically equivalent. The rest ones are not homeomorphic to each other, since a continuous mapping between them cannot be determined. We say that model from Figure II.6.e is topologically equivalent to the unit sphere. Also, in Figure II.6.f we have an object topologically equivalent with the unit disk or the unit square.

In the rest of this thesis we will operate only with objects homeomorphic to either sphere or disk. We denote the surface of the unit sphere by S_0 and we define it as:

$$S_0 = \{p(x, y, z) \in \mathbb{R}^3 \mid \sqrt{x^2 + y^2 + z^2} = 1\} \quad (\text{II.1})$$

Also, we denote the unit disk with D_0 and we define it as:

$$D_0 = \{\varphi(u, v) \in \mathbb{R}^2 \mid \sqrt{u^2 + v^2} = 1\} \quad (\text{II.2})$$

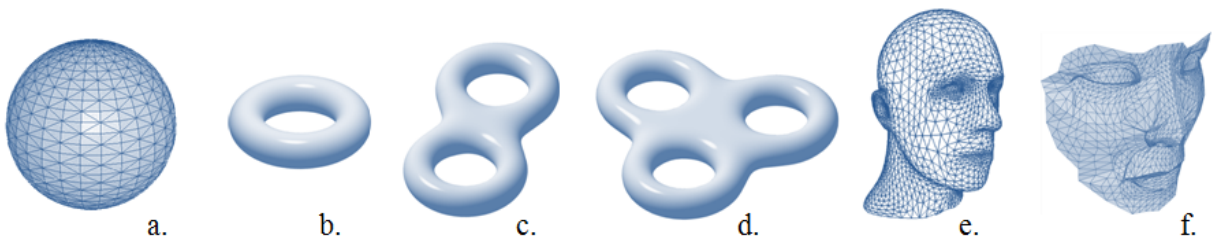


Figure II.6. 3D models with various topologies.

II.2.1.3. Two manifold meshes

In order to introduce the concept of *two manifold mesh*, we have to define first some other terms used in 3D graphics field. We note by $deg(p_i)$, the *degree* or the *valence* of a vertex, which is the number of edges incident to the considered vertex. Next, we can define the *star* of the vertex p_i , denoted by $star(p_i)$, as the submesh which is composed of all the faces containing p_i (i.e., the set of all points adjacent to p_i). The star is also called the *1-ring* of the vertex p_i .

A triangular model is two manifold if the star of any vertex is homeomorphic to a closed disk or a half-disk (at the boundary). This results in three important consequences. First, any edge is incident to at most one or two faces. Second, the triangles incident to a vertex form a closed or an open fan as illustrated in Figure II.7.

Third, the degree of any interior vertex p_i is equal with the number of faces sharing p_i . At the boundary, $deg(p_i)$ is equal with the number of triangles incident to vertex p_i plus one. In Figure II.8 we present some cases of non-manifold meshes.

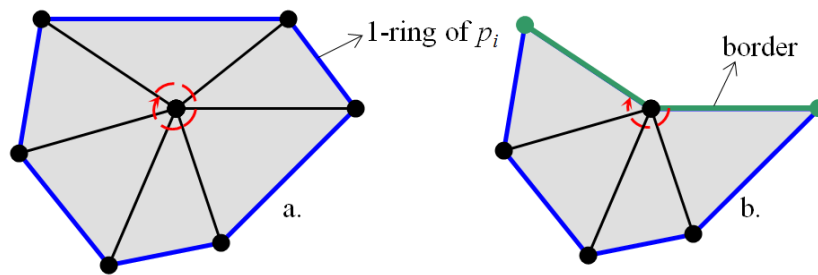


Figure II.7. Two manifold meshes: (a) closed fan; (b) open fan.

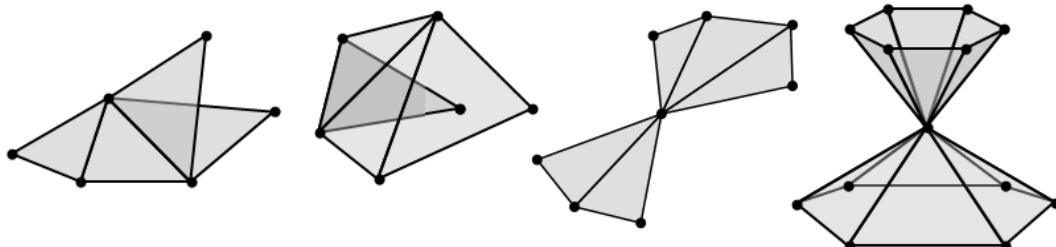


Figure II.8. Non manifold meshes.

II.2.1.4. Orientable meshes

The orientation of a face f is established by the sequence of its vertices. Let p_i , p_j and p_k be the vertices of a triangle. Although faces $f(p_i, p_j, p_k)$ and $f^*(p_i, p_k, p_j)$ coincide in the Euclidian space \mathbb{R}^3 , they have different orientations.

In order to illustrate the concept of orientable meshes, let us consider the example in Figure II.9. We can observe that each face can have two orientations depending on the order in which the vertices are specified. A clockwise or an anticlockwise order in which the vertices are listed defines the directions of the corresponding normal vectors (applying the right hand rule). We say that the orientation of two adjacent triangles is compatible, if the two vertices of the common edge are specified in opposite order. Then, we can define an orientable mesh as a manifold mesh with compatible orientation for any two adjacent faces.

Figure II.9.a illustrates the concept of orientable mesh where the common edge $e(p_i, p_k)$ is traversed in opposite directions in the two neighbor faces, while Figure II.9.b shows a non-orientable mesh where the common edge $e(p_i, p_k)$ is traversed in the same sense.

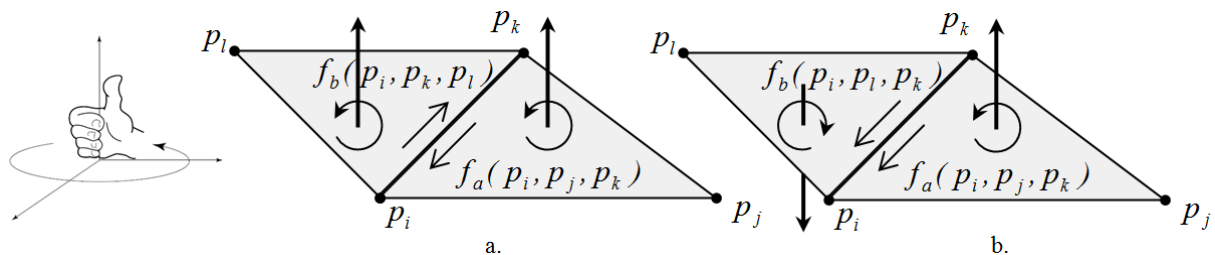


Figure II.9. The concept of orientable mesh: (a) orientable mesh; (b) non orientable mesh.

If a mesh has at least two triangles with different orientations, the mesh is considered non-orientable. In our work, we will consider only manifold and orientable meshes.

II.2.1.5. Closed and open meshes

An edge belonging to a single triangle of the mesh is called a border edge. Border faces are defined as triangles including at least a single border edge. A non border-edge belongs to two different triangles and is called an internal edge. Two faces are said to be e-neighbor faces if they share at least a common edge. A sub-set of mesh faces is connected if between each two component faces can be found a path of successive e-neighbor faces.

A connected mesh is said to be closed if any component edge share exactly two triangles, *i.e.* it not contains any border edge. The typical example of a closed mesh is a triangle mesh that tessellates a sphere (Figure II.6.a). It can be verified if a model is closed or not using the Euler formula (*cf.* Section II.2.1.6). If a mesh is not closed, it is said to be an open mesh (Figure II.6.f). For an open model, the set of border edges is referred to as the boundary of the mesh.

II.2.1.6. Genus of a surface and Euler formula

The concept of genus is introduced to define the type of topology for a surface. Surfaces of the same genus are topologically equivalent (homeomorphic) to each other. Intuitively, the genus of a model can be interpreted as the number of holes of the given object. More rigorously, the genus of a surface is defined as the largest number of non-intersecting simple closed curves that can be drawn on the surface without separating it [Wei10] in multiple connected components. The genus of a sphere is zero since no such curves can be drawn on its surface without separating it. The genus of a torus is 1 since one, and no more than one, such curves can be drawn on its surface without separating it. The models presented in Figure II.6.a, b, c and d are of genus 0, 1, 2 and 3 respectively.

In the middle of the 18th century, Euler discovered a mathematical relation between faces, edges and vertices of a simple polyhedron. The Euler's formula can be expressed as:

$$V - E + F = 2 \quad (\text{II.3})$$

where V denotes the number of vertices, E the number of edges and F the number of faces of a closed polyhedron.

Considering a triangular mesh with V vertices, E edges and F faces, we observe that every face has 3 edges and every edge is shared by two faces thus it follows that:

$$E = \frac{3}{2}F \quad (II.4)$$

Combining the Euler formula (II.3) with equation (II.4) and knowing the number of vertices, we may determine the number of faces and edges in a mesh:

$$F = 2(V - 2), \quad (II.5)$$

$$E = 3(V - 2) \quad (II.6)$$

Relation (II.3) is only valid for closed, manifold, genus 0 meshes, but can be generalized also for meshes with a boundary:

$$V - E + (F + 1) = 2, \quad (II.7)$$

or for meshes with an arbitrary genus:

$$V - E + F = 2(1 - G), \quad (II.8)$$

where G is the genus of the mesh.

II.2.2. File formats for mesh storage

A big amount of specification file formats have been provided in order to store and exchange 3D meshes. Let us recall some popular examples of such 3D storage formats:

- The 3D Object File Format (OFF - developed in 1986 at Digital Equipment Corporation's Workstation Systems Engineering) for the interchange and archiving of 3D objects. OFF is an ASCII-based format and is independent of languages, devices, and operating systems.
- The Wavefront Object Format (OBJ - a geometry definition file format first developed by Wavefront Technologies for its Advanced Visualizer animation package),
- The Stanford University's PoLYgon format (PLY),
- The 3D Studio Max format (3DS - Used by the AutoDesk 3D-Studio and 3D-Studio MAX commercial modeling, rendering and animation package on the PC),
- The SMF(Simple Model Format - which is a subset of the Wavefront OBJ file format)
- The Virtual Reality Modeling Language (VRML), which became an ISO international standard.

Their storage strategies are very similar, based on the face–vertex mesh representation method. First, the vertices positions in the 3D space (sample points with (x, y, z) coordinates) are presented in an unorganized way consisting on the vertex list. Then, the polygon primitives (in the most cases - triangles) are also defined by an unorganized face list.

Each entry of the faces list defines a triangle by the indices of its vertices (indexed by their order of appearance in the vertex list). In other words, any mesh file format will store mainly the geometry and the connectivity of the model. However, additional information can be included, such as color, normal vertices, transparency or texture data.

Due to its early international standardization, the Virtual Reality Modeling Language (VRML file extensions) succeeds to become highly popular in the computer graphics field. VRML has been originally specified based on the Open Inventor API paradigm developed by Silicon Graphics Inc. and has been first proposed to the International Organization for Standardization (ISO) in 1994 by the Web3D Consortium in order to provide a standard technique for modeling 3D interactive experiences over the Web. Its successor (X3D, based on XML) has been approved for standardization in 2005, but has still less success than the second version of VRML (proposed in 1997).

VRML's technology has very broad applicability, including web-based entertainment, distributed visualization, 3D user interfaces to remote web resources, 3D collaborative environments, interactive simulations for education, virtual museums, virtual retail spaces, and more.

In the VRML format, 3D objects are specified in a dedicated node, so-called "Shape". This node type has several attributes, including its material appearance (for lighting modeling) and its geometry. The "geometry" attributes can be valued with pre-defined shape primitive nodes or with an "IndexedFaceSet" node. This latter node has two main attributes which are the coordinates of the sample points ("coord" field, valued with a "Coordinate" node) and the face specification ("coordIndex" valued with an array of vertex integer indices) [Sch98]. Consequently, the VRML format could cover the most basic mesh needs:

- a list of vertices;
- a list of faces;
- a list of materials (texture and color);
- a list of texture coordinates;
- a list of lights (material, description and position).

A standard layout for surface mesh storage of a simple object (*i.e.*, a cube) with VRML v2.0 can be written in the following way:

```
#VRML V2.0 utf8
DirectionalLight {
  ambientIntensity 1
  color            1 1 1
  direction        0 0 -1
  intensity        0
  on               TRUE
}
DEF MATERIAL Material {
  diffuseColor 1 1 1
}
Shape {
  geometry IndexedFaceSet {
    coord Coordinate {
      point [
        # sample point coordinate (x, y, z) list
```



```
        0 0 0
        1 0 0
        ...
    ]
}
texCoord TextureCoordinate {
    point [
        0.1291 0.3485
        0.1706 0.3248
        ...
    ]
}
coordIndex [
# face list: vertex indices (face separator: "-1")
0 1 2 -1
0 1 5 -1
...
]
}
}
```

II.3. CONCLUSIONS

We have presented in this chapter the theoretical background related to 3D mesh representations. We defined the terms of triangular polygonal mesh, which will be considered all over our work. Several requirements, in terms of topological and geometric properties have been identified. In particular, a closed mesh M has to meet the following conditions:

- M is homeomorphic (topologically equivalent) to the sphere;
- M is two manifold;
- M has no border edges or faces;
- M is an orientable mesh;
- the number of vertices, faces and edges of M has to satisfy the Euler formula (II.3) and equations (II.4), (II.5), (II.6).

Finally, we have also presented some features of the Virtual Reality Modeling Language (VRML) standard that we have adopted for 3D mesh representation.

III. AN OVERVIEW OF 3D MESH MORPHING TECHNIQUES

Summary: *This chapter first states the problem of object metamorphosis, enunciating the main principles and necessary steps involved in a morphing method. After a brief synthesis of the 2D image morphing techniques, we provide an overview of the main 3D mesh morphing approaches proposed in the state of the art. The study reveals the necessity of designing morphing techniques able to gradually transform two 3D objects while maintaining aligned the corresponding features of interest. Finally, the morphing framework adopted in our work is here presented.*

III.1. MORPHING IN GENERAL

The word morphing is derived from the word “metamorphosis”, which according with the Oxford Dictionary [All91] has the following meaning:

“A change of the form or nature of a thing or person into a completely different one”

Thereby, in the case of 3D meshes, the term morph can be interpreted as the change of appearance of a graphical object. The morphing process is then defined as the construction of an animated sequence corresponding to the gradual transition between two different objects, so-called source (initial) and target (final) models. The objective of a morphing method is to compute a transformation ensuring a visually pleasant transition between the two, source and target shapes.

Existing professional animation environments, such as 3DS Max or Lightwave, propose some basic morphing techniques. However, such methods cover only partially the aspects that need to be taken into account. In particular, they are able to morph solely meshes with the same topology and number of vertices and thus severely restrict the field of possible applications. Thus, one important objective is to make possible to morph 3D models described by different topologies, numbers of vertices and connectivities.

The concept of morphing is illustrated in Figure III.1 where the leftmost object is morphed into the rightmost. The upper and lower sequences show the metamorphosis in two possible manners. Obviously, there is no unique solution for the morphing process and a set of criteria and evaluation measures/protocols has to be defined in order to validate the various solutions. One important (and ultimate) criterion is the visual quality, in terms of smoothness and fluidity of the obtained transitions.

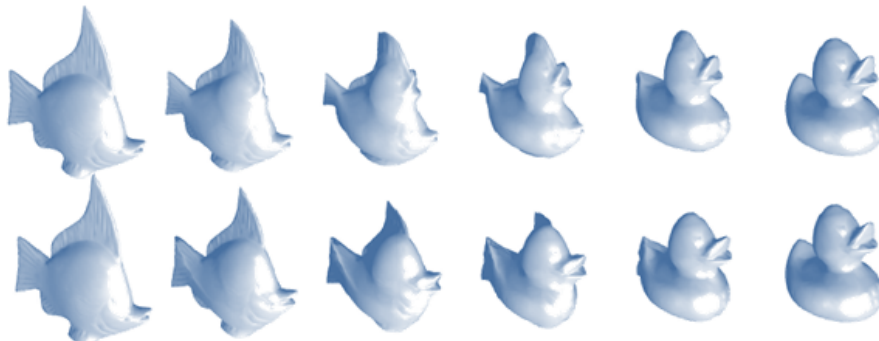


Figure III.1. Two morphing sequences between a fish and a duck model [Ath10].

The next section introduces some useful criteria that have been established to evaluate a valid morphing.

III.2. VISUALLY PLAUSIBLE MORPHING

In order to obtain a “good” morph, the geometric and topological properties of the two, source and target objects during the transformation should be preserved as much as possible. This implies the set-up of a highly smooth transformation process.

From a more rigorous point of view, the main principles that have to be considered for ensuring a visually pleasant shape transformation relate to topology preservation, feature preservation, rigidity preservation, smoothness and monotonicity, as introduced by Gomes *et al.* in [Gom99]. Let us briefly recall them here below:

- *Topology preservation:* Various definitions of topology preserving transformations have been proposed [Kon89], [Lie04], [Sah96]. Two objects have the same topology if they have the same number of connected components and genus. Intuitively, in the case of a morphing process, preserving the topology of both source and target objects requires to ensure that no holes or other artifacts appear during the transformation. Let us note that most of the time, the source and target objects are required to have the same global topological properties in order to enable the morphing process.

- *Feature preservation:* means that the shapes of important features which are present both in the original object and in the target object are preserved during the morphing transition. In particular, this requires a good correspondence between them. For example, in Figure III.1, the first row represents a morphing transition from a fish to a duck, realized with no feature preserving. Here, the different features are completely unaligned, which leads to an unnatural transition sequence. The second row presents a morphing sequence where the mouth of the fish corresponds to the one of the duck and both remain aligned during the transition. Establishing in an automatic manner, pertinent correspondences between features of interest is however a highly challenging task. Some automatic solutions are proposed in [Urt04], [Ath10], [Gil09]. Such techniques are useful in the case of source and target objects with a sufficient number of similar features that can be put effectively into correspondence. However, in a more general framework the human interaction is mandatory. For this reason, the user should be allowed to specify and control the feature correspondences in an interactive manner. Some examples of works that require user interaction are presented in [Ros94], [Ler95], [Zhu09], [Kan98], [Gre98]. The control should be neither time-consuming nor labor-intensive and adapted to user's knowledge and skills. Providing such a control is also a nontrivial task and requires the elaboration and development of appropriate, ergonomic user interfaces with all the necessary interactivity features.

- *Rigidity preservation*: this principle refers to the facts that in certain cases, some metrics should be preserved during the transformation. Typical examples considered in the literature include angles between edges/faces, lengths or convexity measures.
- *Smoothness*: the shape transformation should be fluent, avoiding discontinuities and artifacts. If we view the transformation as a function f that maps vertices from the source mesh onto the target mesh, we have to impose for the f function to have continuous derivatives up to some desired order over the deformation domain. The number of continuous derivatives, will influence the smoothness of the transition function and thus, of the morphing process.
- *Monotonicity*: this principle states that the volumes, areas, or parts of the source shape should change monotonically during the interpolation process. In other words, it is not allowed that parts of the model to decrease (/increase) so that later to increase (/decrease) back. The monotonicity principle makes it possible to avoid local self-interactions of the intermediates meshes obtained.

Jointly satisfying the set of all these constraint is a difficult problem and the existing methods are privileging some of the above-mentioned aspects, depending of the application considered and of the specific morphing effects that are targeted.

Historically, the first morphing techniques of visual entities appeared in the case of static 2D images. Such approaches set the general principles and methodologies useful for morphing of any graphical entities. So, before considering the case of 3D mesh morphing, let us first analyze the traditional techniques proposed in the field of 2D image morphing, which are today mature and extensively used in commercial applications.

III.3. 2D IMAGE MORPHING

Image (or 2D) morphing can be defined as the construction of sequence corresponding to a gradual transition between a source image and a target one. Such techniques have been intensively used to produce visual effects for various entertainment applications. A lot of examples can be provided from the movie industry, starting with the first movie that implemented morphing, “Le Magicien” in 1898, and continuing with films like “Indiana Jones”, “Terminator”, or the more recent one “Transformers”.

A particular case of morphing/interpolation is encountered in the field of medical imaging, in the case of MRI scans. Here, the acquired slices are available at a fixed inter-slice resolution. The distance between such slices is usually higher than the spatial resolution within each slice. For rendering and surface reconstruction, some interpolation between slices is

mandatory. Methods using image morphing to create intermediate images between slices are presented in [Rup94a][Rup94b]. Figure III.2 illustrates such an example, where the stack of original images completed with those created by morphing techniques looks much more smooth and realistic.

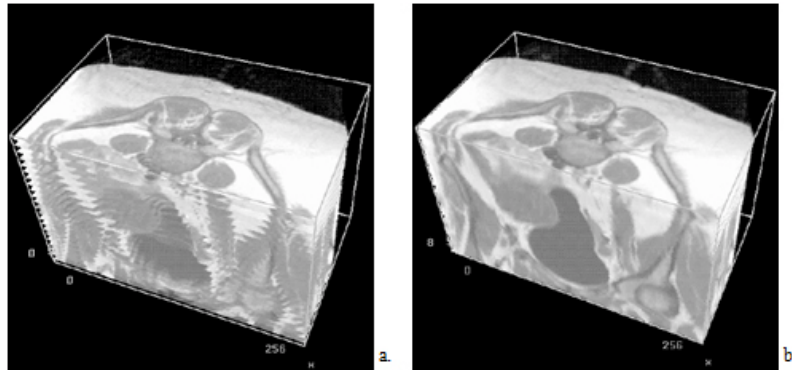


Figure III.2. Volume representation of a stack of images: (a) without, (b) with interpolated images [Rup94a].

Generally, the morphing effects are obtained with the help of the well known cross dissolve or fading techniques which permit to achieve a smooth change of an image content (*i.e.* texture and/or color) from source to target frames. The color of each image pixel is interpolated over time from the source image value to the corresponding target image value. Most often, linear interpolation is utilized. This process is called cross-dissolve interpolation. Unfortunately, such naïve approaches do not allow obtaining a pleasant visual effect, because the two images (source and target) overlap gradually without any preliminary alignment of the main features of interest present in the content. This problem is illustrated in Figure III.3.a. Here, on the intermediate images obtained we can notice the presence of overlapped features (*e.g.*, nose, eyes, mouth...) from both source and target models, which is visually uncomfortable.

This example shows the interest of applying a global transformation between source and target images that can be able to align the main features involved. Such an alignment can be achieved with the help of the so-called *warping methods* [Gom99]. Image warping applies 2D geometric transforms to the images in order to obtain a geometric alignment of their features of interest, followed then by a color interpolation to blend their corresponding colors.

The effectiveness of introducing warping methods in the morphing process is illustrated in Figure III.3.b. Here, image warping is combined with the classical cross dissolving. As the morphing process progresses, the source image is gradually warped into the destination image and faded out, while the target is gradually warped into the early picture and faded in.

In this way, the first images in the morph sequence will be similar to the source, the middle images of the sequence will correspond to an average of the two source and target models and the last images will be similar to the target picture.



Figure III.3. Image Morphing: (a) Cross-dissolve; (b) Warping and cross dissolve [Gom99].

We can observe that the obtained sequence is visually more pleasant than in the case of a simple cross-dissolve, the transition between corresponding features being more natural. We also note that the middle images strongly determine the overall quality of the morphing process.

Creating a morph using the deformation technique involves specifying a warp, *i.e.* a bijective (and so, invertible) transform of the source image into the target one. Some examples of basic geometric transforms that can be jointly used for warping purposes are illustrated in Figure III.4, for a checker board image.

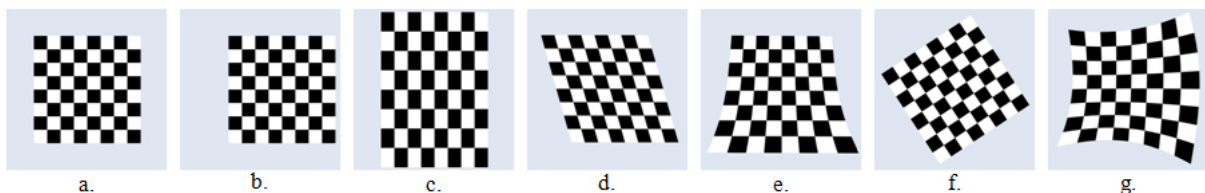


Figure III.4. Different types of warps: (a) original image; (b) shift to the right; (c) scaling in the vertical direction; (d) shear; (e) scaling in the horizontal direction; (f) rotation; (g) quadratic.

The issue of constructing a smooth warping field between two different images has been extensively studied in the rich literature dedicated to the subject. Different approaches model the image with the help of a 2D elastic mesh and are known under the name of mesh

warping techniques [Wol90], [Sho03], [Kan97]. Multilevel free-form deformations [Lee95], field morphing [Bei92], [Nis93], radial basis functions [Edg03], or energy minimization [Lee96] can also be used to achieve 2D morphing.

The warping-based morphing technique proposed in [Wol90], uses two 2D meshes M_S and M_T which are respectively associated with the source I_S and the target I_T images. The two meshes share the same connectivity and are thus defined with the same number of faces and vertices. A set of control features is determined starting from each source and target images and then associated with each mesh. The meshes are used to define the spatial transform, by linear interpolation, mapping all points in source image onto target image, while keeping initial and transformed meshes topologically equivalent, *i.e.*, no folding, self-intersection or discontinuities are permitted. Furthermore, for simplicity, the meshes are constrained to have fixed boundaries.

Figure III.5 illustrates the morphing process in the case of two faces. Feature points correspond here to eyes, nose, cheeks and lips. In the top row, the mesh M_S is deformed to mesh M_T , producing an intermediate mesh M for each frame. These meshes are used to warp the source image into increasingly deformed images, thereby deforming I_S from its original state to those defined by the intermediate meshes. The same process is presented in reverse order in the bottom row of Figure III.5, where I_T is shown deforming from its original state.

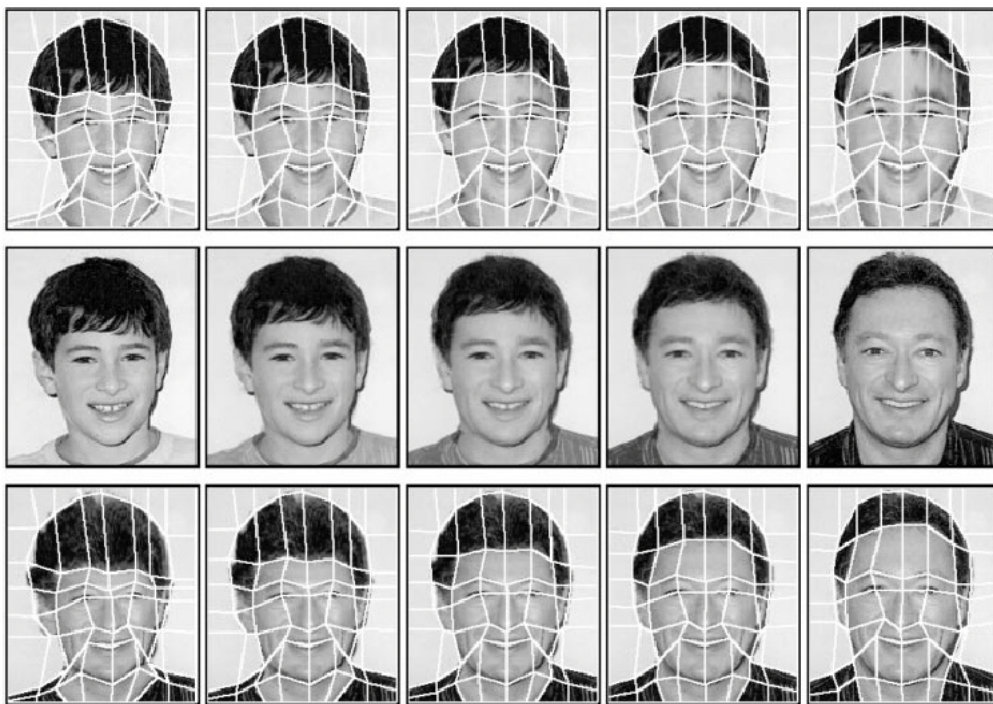


Figure III.5. Image mesh warping [Wol98].

The purpose of this procedure is to establish the feature alignment between I_S and I_T as they both deform to some intermediate state, producing the pairs images shown in the top and

bottom rows, respectively. Once the alignment is performed, the cross-dissolve between successive image pairs is applied, yielding the morphing sequence illustrated in the middle row.

The field morphing technique proposed by Beier and Neely [Bei92] simplifies the task of feature specification by establishing a set of line segments in both images. For example, rather than requiring the correspondence points to lie on a mesh, line segments can be drawn along the mouth, nose, eyes, and cheeks of the source and target images.

Therefore only a set of key features needs to be provided. Although this approach simplifies the specification of feature correspondence, the generation of the warping transform becomes more complex. This is due to the fact that all feature pairs must be considered before the mapping of each source point is known.

The algorithm warps only the set of pixels (lines) specified by user, moving them exactly where the user want and everything else is blended smoothly based on those positions. This approach can lead to unexpected displacement between the control lines, which manifest by hiding parts of the image or showing them up in some other regions of the interpolated picture. Additional control line pairs must sometimes be supplied to counter the ill effects of the previous set.

Edge and Maddock [Edg03] propose a more general form for feature specification that permits to specify landmarks on both images that consist of points, lines and curves. The authors use then radial basis functions [Dyn89] to put in correspondence the feature points, deforming the two input images accordingly.

The above-described techniques do not guarantee the one-to-one property of the generated warp functions. When a deformation is applied to an image, the one-to-one property prevents the warped image from folding-over. Lee *et al.* [Lee96] propose an energy minimization approach in order to obtain a one-to-one warp function. The technique allows feature specification primitives such as points, segments and curves which are sampled and reduced to a set of points. Such points are then used to generate a warp function that is interpreted as a 2D deformation of a rectangular plate. The constraints for a one-to-one warp are represented in terms of energy minimization. The technique generates natural deformations since it is based on physically meaningful energy terms. The main limitation of the method is related to its high computational cost.

A simpler and faster method is presented in [Lee95]. Multilevel free-form deformation (MFFD) is here applied across a hierarchy of control lattices in order to generate one-to-one and C^2 -continuous warp function. In particular, deformations are obtained from positional

constraints by introducing the MFFD as an extension to free-form deformation. Lee *et al.* use the bivariate cubic B-spline tensor product to define the free form deformation function. A new direct manipulation technique for free form deformations, based on 2D B-spline approximation, is applied to a hierarchy of control lattices to exactly satisfy the positional constraints. To guarantee the one-to-one property of a warp, a sufficient constraint for a 2D cubic B-spline surface to be one-to-one is presented. The MFFD generates C^2 -continuous and one-to-one warps that yield fluid image morphing sequences.

However, the problem of the two dimensional images is that they do not take into account any information related to the shape of the objects present in the scenes. This results in a number of shortcomings in the transformation processes, due mainly to occlusion phenomena: since the 2D images often represent projections of virtual 3D scenes, the intermediate results of a 2D image morphing process may not correspond to the rendering of a morphed 3D scene. Thus, in the case of 2D morphing, pixel values are simply interpolated between source and target images. On the other hand; in the case of 3D morphing, for each intermediate stage a complete 3D representation of the scenes/shape is determined. Then, the intermediate shape is rendered and the associated photometric information is represented according to the 3D shape, lights and camera position.

Thus, 3D representation of objects allows animation to be independent of any projection transformation and user to have the flexibility to change the position of camera during a certain transition, so that the morphing can be observe from different points of view.

III.4. 3D OJECT MORPHING

Let us first briefly overview the volume-based morphing approaches, which represent in a certain manner the direct extension of 2D image morphing techniques.

III.4.1. 3D volume based morphing

Depending of the way the control features required for guiding the morphing process is used, the following three families of volume-based morphing approaches are identified:

- cross dissolving - no control features are required in this case;
- field morphing - where control features are used to specify coordinate mappings;
- mesh warping - where control features define both volume subdivisions and coordinate mappings.

Analogously to the case of 2D images, the cross dissolving technique involves a direct interpolation of the source and target 3D representations, without any geometric deformation

of the corresponding volumes (Figure III.6). The simplest cross dissolving method is a linear interpolation between source and target volumes [Pay92]. Without surprise, such a technique is too simplistic to yield satisfactory results.



Figure III.6. 3D cross dissolve based morphing [Efr11].

To enhance the smoothness of the in-between volumes, a Fourier transform may be exploited within the framework of a non-linear interpolation scheme [Hug92]. In this case, the high frequencies of the source model are gradually removed during the morphing while the low frequencies are interpolated to those of the target and the high frequencies of the second model are gradually added in.

Wavelet transform, which provide a multi-resolution space-frequency representation, can also be used in morphing purposes as proposed in [He94] (Figure III.7). The idea is to decompose the models into a set of frequency bands, apply smooth interpolation between the volumes to each band, and then reconstruct the morphed volume. Furthermore, the decomposition and reconstruction processes are accomplished in a multiresolution manner so that high frequency distortion can be adjusted to the desired level.

Although they are easy to use and fast to run, such methods have difficulties in producing high quality results in most cases, especially when the transformations between two volumes involve scaling or rotation.

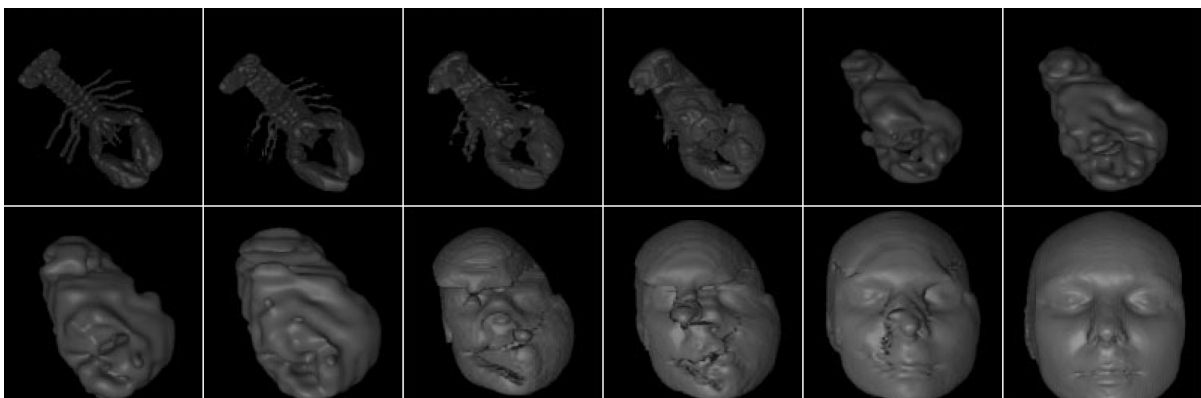


Figure III.7. Wavelet Domain Volume Morphing[He94].

In the case of field morphing approaches, a set of control features is used to specify key regions of interest of the volumetric data that are exploited for performing in-between mappings. Such control features are represented as sets of points (point field)[Rup94b], lines

(line field) [Che95] or disks (disk field)[Che99] defined in the 3D volume. Most often, a user interaction is required to specify such control features. 3D feature-based morphing techniques naturally extend 2D image methods were they have demonstrated their flexibility and controllability in metamorphoses [Bei92].

Given a starting volume V_a and a final volume V_b , both represented as a collection of voxels organized in the form of a three dimensional grid, the morphing process generates a sequence of in-between volumes V_n which represent a smooth transformation from V_a to V_b (Figure III.8).

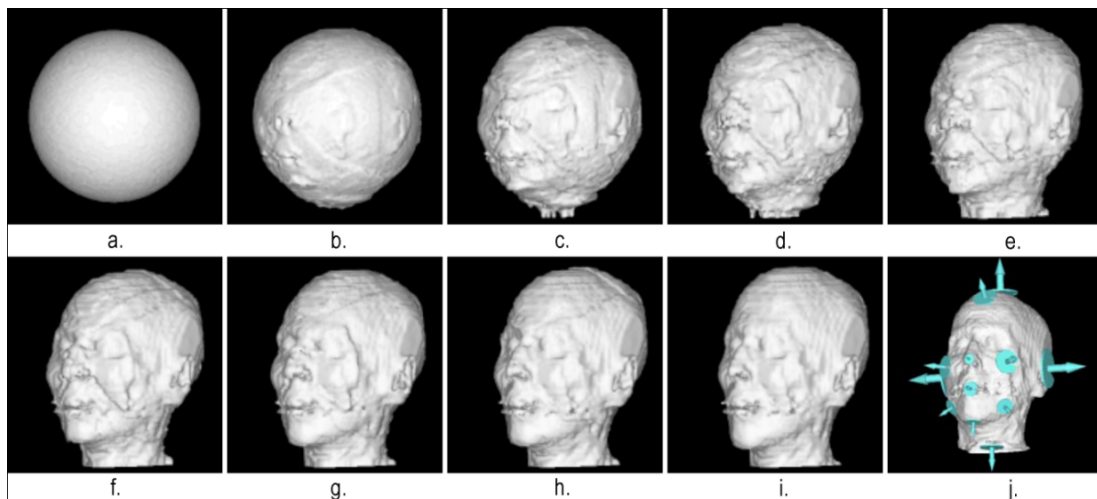


Figure III.8. A volumetric field morphing sequence: (a) source volume; (b)-(h) in-between volumes; (i) target volume; (j) disk fields that control the sphere-to-head morphing[Che99].

For the n^{th} in-between volume V_n , an intermediate control dataset is firstly obtained by linear interpolation of the original control features pairs. Thus, under the influence of the new control dataset, two deformed volumes V_{na} and V_{nb} are obtained corresponding to V_a and V_b . The in-between volume V_n is then simply an interpolation of those two volumes using a cross dissolving technique as previously presented.

A method using a combination of different fields, which include points, lines and boxes, has been reported in [Ler95] and [Man99]. The naive extensions of point and line fields suffer from the inability to specify arbitrary 3D coordinate mappings. The problem can be solved by introducing a supplementary vector in each line field (or two vectors in each point field) [Che99], or by solving a set of error functions [Rup94b]. However, in contrast with the 2D case, such fields are generally difficult to define and manipulate.

The mesh warping techniques are also based on a control feature set and extend the method proposed in [Wol98] for 2D images, where the control features are based on a planar subdivision associated to each image, typically a parametric grid or a triangular mesh. Two volume warping methods presented in [Che95] extrapolate such an approach for the case of 3D volumes. Given a set of points, a volume subdivision can be automatically generated, for

instance, by 3D triangulation. In mesh warping, the distortion is constrained by individual elements, and it is therefore relatively easier to achieve a desired transformation without causing “ghost shadows” effects [Bei92] (*i.e.* no triangles overlapping) (Figure III.9).

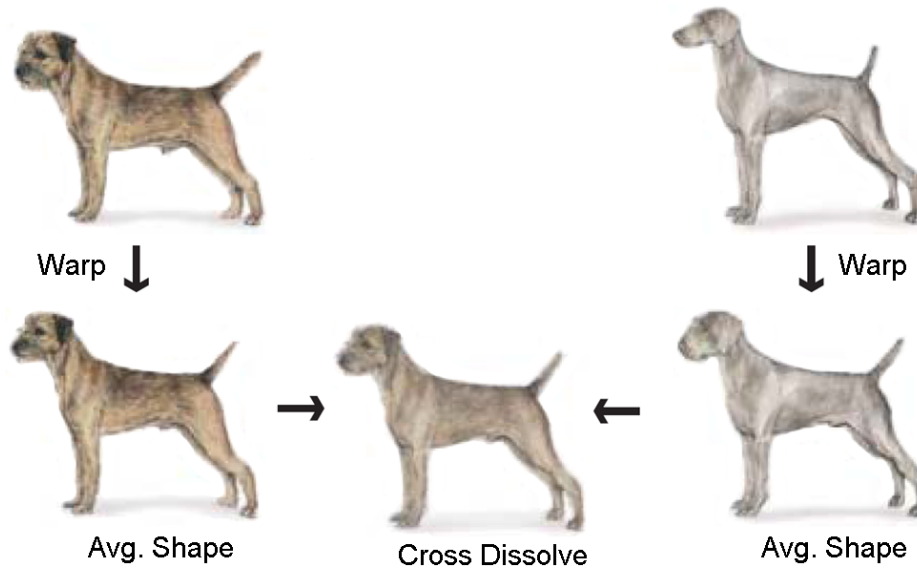


Figure III.9. Morphing by mesh warping[Efr11].

Volumes V_a and V_b are first partitioned respectively by two spatial subdivisions that are of an equal number of elements. Deformed subdivisions are then obtained for in-between volumes by interpolation. A voxel in an in-between volume is mapped onto a voxel in each of V_a and V_b . The values of voxels in the in-between volume are determined by linearly interpolating those of the corresponding voxels in V_a and V_b .

However, in most cases, both field morphing and warping-based methods require a high number of control elements, in order to avoid undesirable visual artifacts [Che95]. In addition, the manipulation of 3D subdivisions with dedicated user interfaces cannot be handled in a straightforward manner.

The volume-based techniques offer the advantage of being less sensitive to different object topologies when compared with the surface based morphing techniques, since the objects are here defined on a fixed, voxelized topology.

However, volumetric representations are useful in the case of some specific applications (*i.e.*, medical imaging), but less used for general public applications, because of the bandwidth and storage capabilities required. In addition, the corresponding volume morphing approaches suffer in general from their high computational complexity.

In the rest of this thesis we will focus on surface representation techniques and in particular on 3D meshes.

III.4.2. 3D mesh morphing

Intensive research has been dedicated to the issue of 3D mesh morphing techniques, as testifies the rich literature on this subject.

III.4.2.1. Problem statement

Triangular 3D meshes are discrete 3D object representations that offer the advantage of being able to represent a large variety of complex geometries. In contrast with 2D images or 3D volumes, which are defined on a fixed topology (*i.e.*, 2D or 3D lattices), 3D meshes may exhibit significant differences in terms of topological properties. Thus, they can present different numbers of vertices/faces and are often defined on highly irregular connectivities.

Because of such specificities, an initial stage is here required, which consists of establishing a correspondence between the two source and target 3D discrete surfaces defined by the meshes. Such a correspondence cannot be directly defined, because of the complexity of the topological and geometric information involved. Instead, the correspondence is achieved in an indirect manner with the help of parameterization techniques, which consists of establishing a bijective mapping between the mesh surface and a common 2D domain.

The parameterization can be defined as a map $\Omega: M \rightarrow D$ of a 3D model M to a parametric domain D . Most often, the domain D is either the unit disc (*planar parameterization*), or the unit sphere (*spherical parameterization*) (Figure III.10).

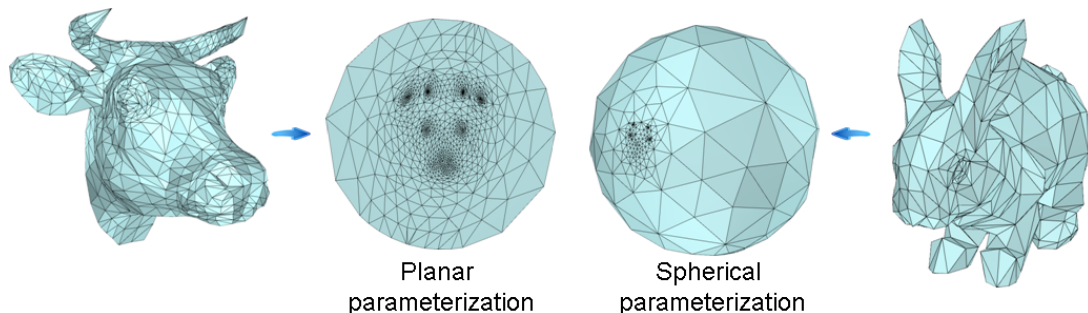


Figure III.10. Examples of 3D mesh parameterizations.

The planar parameterization is useful in the case of 3D meshes that define an open surface with a unique connected component and border. Spherical parameterizations are necessary in the case of closed, connected 3D surfaces with genus-0 topology. Other parametric domains (*e.g.*, torus, atlases, object-dependent ...) can also be used for objects with more complex topologies.

The parameterization represents a mandatory and important step in the morphing process and they condition the overall quality of the metamorphosis. Let us also note that such algorithms are generally time consuming and intractable in the case of complex 3D models

with hundreds of thousands of vertices. In this case, mesh simplification procedures may be required in order to reduce the computational complexity.

However, parameterizing the source and target meshes on a common parametric domain does not entirely solve the correspondence problem. The mesh geometry (*i.e.*, position of the mesh vertices in the 3D space) can be defined in arbitrary coordinate systems. Thus, preliminary normalization and alignment processes are required. Most often, such techniques exploit Principal Component Analysis (PCA) in order to define coordinate systems linked to the considered geometries that need to be aligned.

Moreover, as in the case of 2D images, a *feature alignment process* is necessary in order to guarantee a successful morphing process. This comes to (1) define a set of features of interest on both source and target models and (2) apply a warping/deformation of the parametric domain in order to guarantee that the parametric position of the corresponding features are as closed as possible for both models. We speak in this case of *overlaid parameterizations* (Figure III.11).

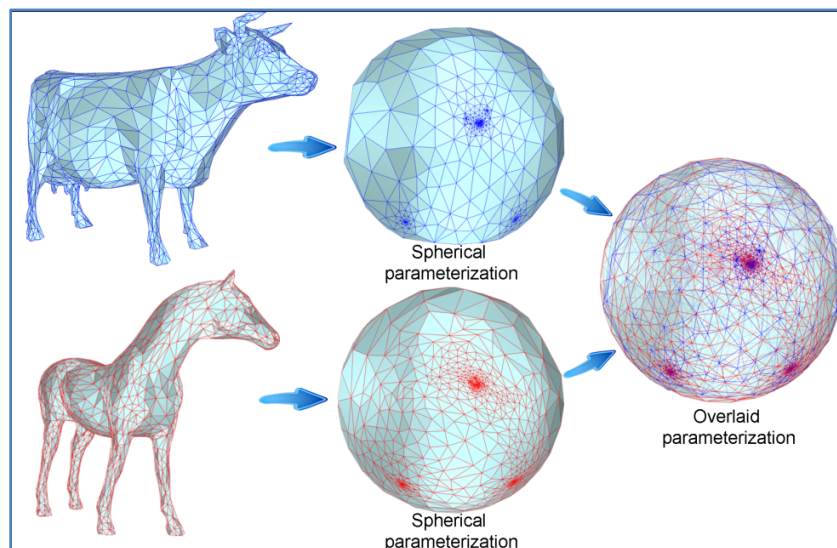


Figure III.11. Overlaid parameterization of two spherical mappings.

The features of interest are in general sets of points, lines, curves, regions, defined over the models to be morphed. They correspond to intuitive, semantic morphological characteristics. As examples, in the case of morphing models of human faces such features may correspond to the regions of eyes, nose, mouth and ears on both objects. In the case of morphing models of animals they might correspond to the regions of limbs, tails, heads...

In order to illustrate the necessity of specifying a number of correspondent features and to maintain them aligned through the transformation sequence, Figure III.12 presents an example of two different morphing sequences between two models of pigs (a young and an adult one). In the upper row, no features were specified and the resulting morph is unacceptable (8 legs are appearing in the middle models). The lower row of Figure III.12

shows a morph produced with a set of features (ears, eyes, hoofs, and the tail) put into correspondence. The result is visually more realistic.

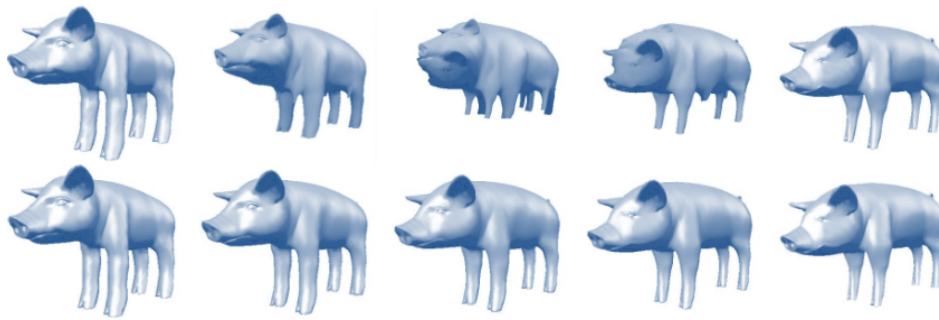


Figure III.12. Morphing between the models of a young pig and a grown-up pig. In the upper row, no feature is specified, which leads to unpleasant effects (8 legs). In the lower row, the eyes, ears, legs, and the tail are put into correspondence yielding a more natural transformation [Ale02].

We have to note that the need of feature alignment becomes more obvious in the case where the models are from the same semantic category. This is because the user has a strong *a priori* expectation of that transformation, and expects that common features of the models (head, legs, tails...) to be preserved.

Let us also note that establishing correspondences between features of interest requires, in general, a significant user interaction. Even some attempts to automate the process have been proposed in the case of some specific object classes [Urt04][Ath10], feature alignment need the development of appropriate user interfaces where features can be specified, selected and put into correspondence in an ergonomic manner.

Once the source and target models are parameterized and aligned with respect to their corresponding features of interest, the final step necessary in the morphing process is the interpolation between objects. This can be done simply by determining the trajectory of the corresponding vertices on the representation obtained in the previous step. At the moment $t = 0$ ($t = 1$) the vertex positions with respect to the source (resp. target) object are known. The simplest way to interpolate between these points is a linear interpolation.

Most of the morphing approaches are based on linear interpolation, but according with [Ale02] this works well only for objects which are rather similar and oriented in the same direction. Linear interpolation of vertices can lead to undesirable effects such as shortening the boundary parts during the transition or self intersections. An interpolation of higher degree is also possible. It yields smoother vertex trajectories, but on the other side it requires some additional information. An interesting idea is introduced in [Gre99], where the user is allowed to specify tangent vectors to define a path for some specific vertices. The modified trajectory is then propagated to the neighboring vertices. By a proper tangent vector specification some cases of self-intersection can be avoided.

The pipeline of a generic 3D mesh morphing scheme, including the various phases involved, is presented in Figure III.13.

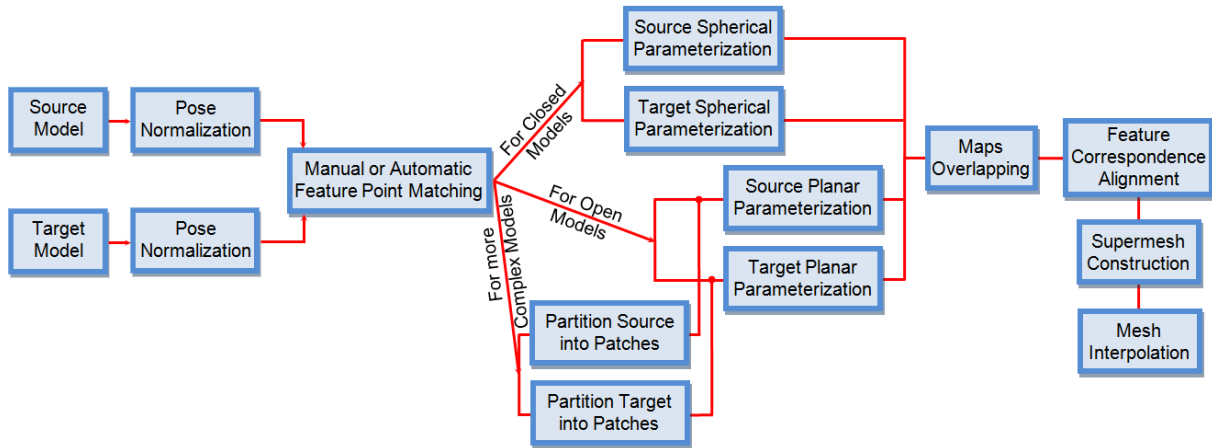


Figure III.13. Generic 3D mesh morphing scheme.

Let us now analyze the solutions to each of the phases involved in a morphing process by the various 3D morphing techniques proposed in the literature.

III.4.2.2. 3D mesh morphing techniques: state of the art

Let us first mention the approach proposed by Kanai *et al.* [Kan98] which are among the first who used the harmonic mapping method in morphing of arbitrary triangular meshes with a topology equivalent to a sphere or to a disk.

For open models the method follows the next principle. First, the user has to specify a boundary loop for each object together with a boundary control vertex, which allows the models alignment. Then, each mesh is embedded into a planar unit disk with the help of harmonic maps [Eck95].

The boundary vertices are mapped into the unit disc border, such that angle formed by two successive vertices and the domain center point is proportional with the arc length determined by the considered vertices. The remaining (interior) vertices are mapped into the interior of the unit disc by minimizing the total energy function defined in equation (III.1):

$$E_{harm}(p) = \frac{1}{2} \sum_{\{i,j\} \in Edges(H)} k_{i,j} \{ \| p_i - p_j \| \}^2 \quad (III.1)$$

where i, j denotes the indices of two adjacent vertices, p_i, p_j represent their geometric position into the unit circle H and p is the set of all interior vertices p_i . $Edges(H)$ denotes the set of edges in H and $\{i, j\}$ is an edge connecting vertices i and j . $k_{i,j}$ is a weight associated to each interior edge $\{i, j\}$, and defined as described in relation (III.2).

$$k_{i,j} = \frac{l_{i,k_1}^2 + l_{j,k_1}^2 - l_{i,j}^2}{A_{i,j,k_1}} + \frac{l_{i,k_2}^2 + l_{j,k_2}^2 - l_{i,j}^2}{A_{i,j,k_2}} \quad (\text{III.2})$$

where $l_{i,j}$ denotes the length of edge $\{i, j\}$ and A_{i,j,k_1} , A_{i,j,k_2} - the areas of adjacent triangles.

Once the harmonic maps (H_S and H_T) of both source (M_S) and target (M_T) 3D meshes are computed, a new object, denoted by H_M is created by overlapping and merging H_S and H_T . H_M , also called supermesh or metamesh, shares the connectivity of both original models and defines one-to-one correspondence between each position of M_S to that of M_T . Kanai *et al.* [Kan98] starts the construction of the supermesh by rotating H_S around the center of the unit disk so that a given boundary control vertex from H_S , selected by user, becomes coincident with the one specified for the target mesh H_T (Figure III.14).

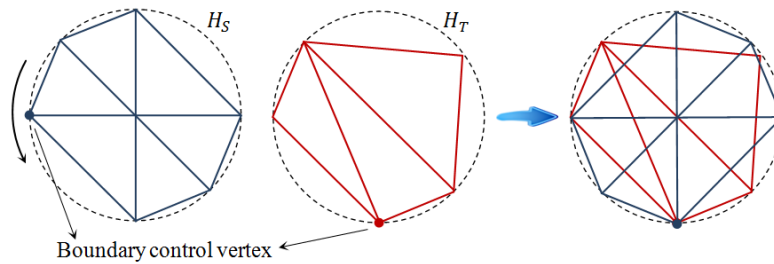


Figure III.14. Maps overlapping and boundary control vertex alignment.

Next, the corresponding 3D positions of the source vertices relative to the target model are computed. First, to determine the 3D position at M_T for each vertex p^S from H_S , the triangle in H_T that includes p^S is established. If p^S is included in a face $f(p_a, p_b, p_c)$ of H_T , the barycentric coordinates (α, β, γ) are computed relative to f . Using this coordinates a corresponding 3D position p^T of vertex p^S in M_T is computed as follows:

$$p^T = \alpha p_a^T + \beta p_b^T + \gamma p_c^T \quad (\text{III.3})$$

where $\alpha + \beta + \gamma = 1$. In a similar way, for each vertex in H_T its corresponding 3D position in the source mesh M_S is calculated. Then, an edge-to-edge intersection between the connectivities of H_S and H_T is computed. For a given pair of intersecting edges, both edges are adequately divided and the mesh is re-triangularized. First, for each vertex, the incident edges are sorted in counterclockwise order. A new face $f(p_i, p_j, p_k)$ is generated using two continuous edges $e(p_i, p_j)$ and $e(p_j, p_k)$ if there is no edge between p_i and p_k . This operation continues until all H_M edges have two adjacent faces. Figure III.15 illustrates the mesh merging process.

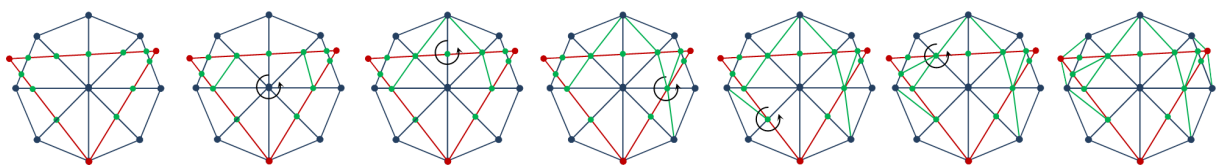


Figure III.15. Supermesh construction.

The resulting super-mesh has a total number of vertices equal to $N_{M_1} + N_{M_2} + N_{Int}$, where N_{M_1} and N_{M_2} are the number of vertices in source and target respectively, and N_{Int} is the number of edge intersections. This will obviously leads to huge meshes in the case of objects described by a large number of faces/vertices. A second drawback of the method consists in the fact that excepting the boundary loop and the boundary control vertex no other feature vertex is set up. Thus, the feature preservation principle cannot be effectively satisfied.

In order to overcome such limitations, an extension of the method is proposed in [Kan00]. Here, the user is allowed to select multiple feature points in the two meshes. Based on this set of vertices, the models are cut into correspondent patches (also called *tiles*) such that the control points remain on the boundaries. By allocating such corresponding vertices to the boundaries of the tiles, the vertex correspondence is satisfied automatically. Next, the mesh patches are individually parameterized into the plane and the supermesh is constructed in a similar way as presented in [Kan98].

Let us note that the specification of the feature vertices and the way the models are cut into patches have to be performed manually, which is poorly intuitive for the user. However, the quality and precision of this process has a great impact on the resulting correspondence.

Finally, in order to interpolate between the source and target model a linear interpolation technique is applied.

Since closed, genus-0 3D meshes are topologically equivalent to a sphere, Alexa *et. al* [Ale00], propose to solve the correspondence problem by a mapping the meshes into the spherical domain. Authors extend the straight-line embedding algorithm proposed by [Tut63] for planar mappings, to the case of spherical parameterizations. This transforms the problem into a nonlinear one, but which can be still solved through a relaxation process. The algorithm iteratively places each vertex at the center of its neighbors and then projects it to the unit sphere. In order to avoid the triangles overlapping or mesh to collapse into a single point, Alexa defines sets of anchor points in the parametric domain which changes at different times during the relaxation process. However, the final embedding is not guaranteed to be valid in all cases.

Based on the feature pair vertices specified manually by user in both source and target models, the problem of feature alignment is established in the parametric domain with the help of a mesh deformation technique based on radial basis functions (RBF) [Ara95]. Given a vertex p that should move to p' the transformation f is defined as:

$$f(x) = \begin{cases} x + (d - \|x - p\|)(p' - p) & \text{if } \|x - p\| < d \\ x & \text{if } \|x - p\| \geq d \end{cases} \quad (\text{III.4})$$

The pair of vertices p and p' are iteratively displaced to the same position on the unit sphere through an iterative process. In the same time, the set of vertices included in the radius of influence d of the considered RBF (equation (III.4)) are also displaced. The amount of this displacement is less than the displacement of the vertex moved, though. However, if the control vertices are forced to reach their final position, the resulting mapping could be invalid. Thus, the features are displaced only as far as their movement did not introduce any foldover.

In order to be able to transform from one object to another, a supermesh is constructed by overlapping the spherical embedding of the two models. Here, the problem of edge-to-edge intersection transforms into a problem of arc-to-arc intersection. Finally, the morphing sequence is obtained using a linear interpolation scheme.

Lee *et al.* [Lee99] employ a multiresolution analysis in order to solve the correspondence problem by generating coarse, simplified models of the two input meshes which are used as base domains (Figure III.16).

Here, MAPS (Multiresolution Adaptive Parameterization of Surfaces) [Lee98] parameterizations of the source and target objects are first constructed. The MAPS algorithm uses a coarse mesh built through successive removal of a maximally independent set of vertices, followed by re-triangulation of the resulting holes. A set of feature points specified by the user is here needed. Such feature points are never removed in the simplification process, thus guaranteeing that they are included in the base domain. Let us note that in addition to control vertices, user can specify also lines of correspondence (*e.g.*, set of edges) to define similar features in the two models. In this case, the parameterization maps all points of the original feature line to a sequence of edges (possibly one) in the base domain.

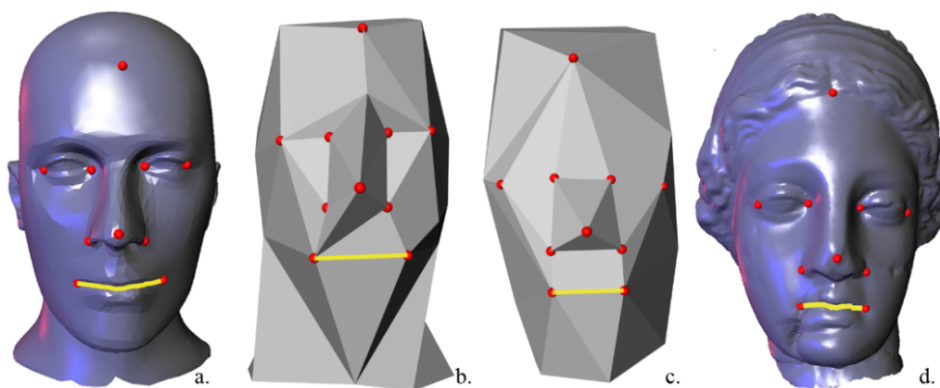


Figure III.16. (a) Source model; (b) the base domain of the source model; (c) target model; (d) the base domain of the target model; [Lee99]

After the base domains are aligned, semiautomatic or manually (if the objects are significantly different from each other), the source map is projected into the target base domain. The initial projection is improved through an iterative relaxation procedure similar to

the technique introduced in [Tur92]. The two final parameterizations of the source and target meshes are obtained using a harmonic map approach in the base domain. Then the meta-mesh including the whole set of source and target vertices is constructed.

The method can morph between relatively dissimilar objects with higher genus manifold (but which share the same genus). The positions of intermediate meshes in the morph sequence are computed based on a linear interpolation. However, the user may have the possibility to control the acceleration of the process or to morph first certain regions before others. This property is illustrated in Figure III.17, where the hair of the character appears before the face. Other attributes such as normal, texture and color information between the source and the target can also be interpolated.

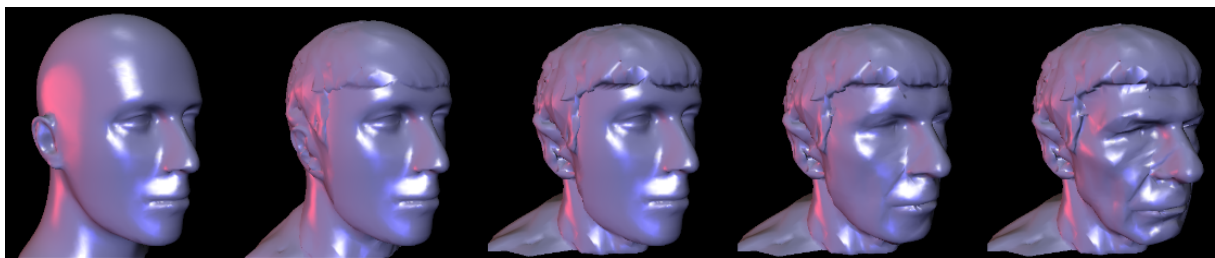


Figure III.17. Morph sequence depending on location [Lee99].

Another multiresolution mesh morphing approach is the MIMesh (*Multiresolution Interpolation Mesh*) technique proposed by Michikawa *et al.* in [Mic01]. First, a base interpolation mesh M^0 is manually created by the user from the input meshes M_S and M_T , which are partitioned into several patches according to the faces of the base interpolation mesh in a similar way as proposed in [Kan00].

Each patch is then parameterized in the planar domain, using a shape preserving mapping algorithm [Flo97], in order to assign a 2D parameter value to each vertex of original models. Next, a subdivision fitting scheme, inspired from the remeshing technique of Guskov *et al.* [Gus00], is applied to create hierarchical interpolation meshes $M^1 \dots M^n$, where n denotes the number of subdivision levels. The method is illustrated in Figure III.18.

The multiresolution interpolation mesh has a semi-regular mesh structure obtained by successive 4-to-1 triangle splits of the base interpolation mesh and only approximates the input models. To achieve the desired approximation accuracy, the number of refinement steps should be adapted to the local geometric complexity of the models. Even so, sharp features of the models cannot be recovered perfectly. In order to overcome this drawback, the authors propose several enhancements. Notably, they propose to exploit a so-called normal map, which corresponds to an image that stores information related to the mesh surface normals (Figure III.19).

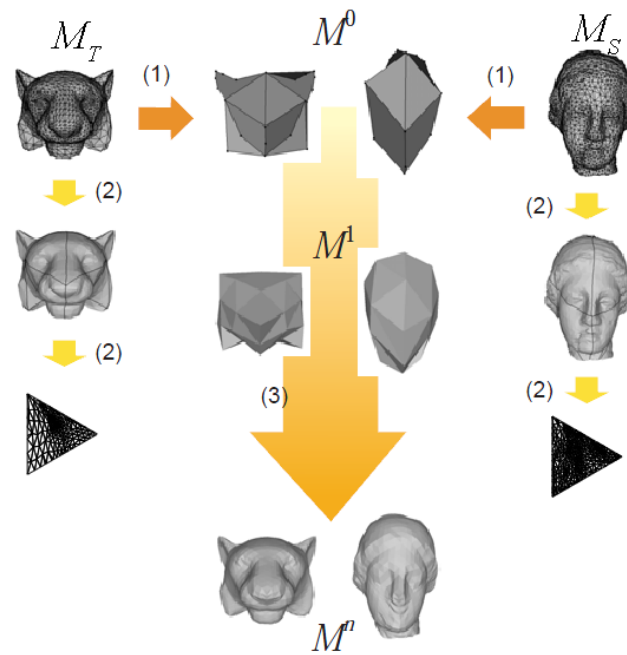


Figure III.18. MIMesh algorithm: (1) Base interpolation mesh construction; (2) Mesh interpolation and parameterization; (3) Subdivision fitting scheme [Mic01].

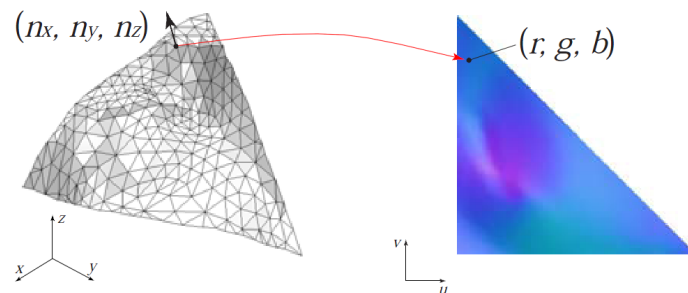


Figure III.19. Normal Map [Mic01].

For each vertex the normal vector $n(n_x, n_y, n_z)$ is computed and for each normal coordinate a value from 0 to 255 is assigned to, accordingly to its magnitude in order to create a normal map represented as RGB image. Pixels in the image map that do not correspond to any vertex, are computed based on barycentric interpolation. Thus, the 3D morphing algorithm interpolates not only between geometries, but also between two normal maps. Figure III.20 presents a visual comparison between a classical MIMesh approximation and the result obtained when applying normal maps.

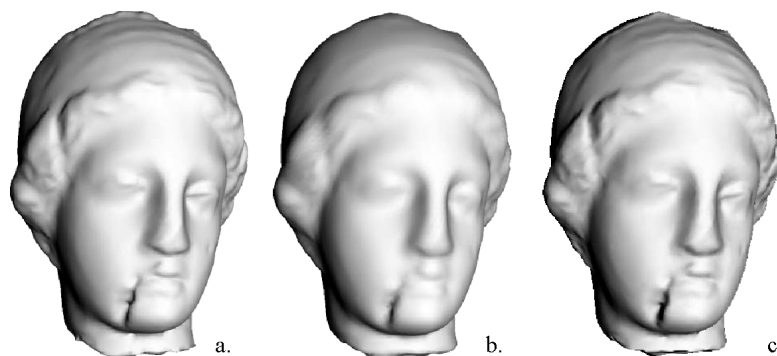


Figure III.20. MIMesh approximation: (a) input model; (b) interpolated mesh without normal maps; (c) interpolated mesh using normal maps [Mic01].

Michikawa *et al.* in [Mic01] present also a non-uniform mesh interpolation technique that permits to locally morph first certain features or regions of interest. However, the drawback when morphing locally arises from the fact that corresponding features might not have the same position in the two models and thus the interpolation can lead to unpleasant artifacts. This effect is illustrated in Figure III.21 where only the nose is morphed.

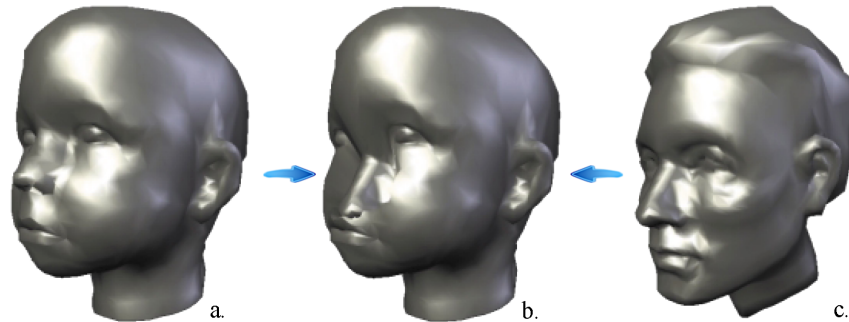


Figure III.21. Local morphing: (a) source model; (b) locally interpolated mesh; (c) target model [Ale02].

A solution to this problem consists of performing the interpolation in different spaces than the initial coordinate system. In [Ale01] and later in [Ale03], Alexa use a rather simple scheme, which represent the vertices in Laplacian coordinates. The Laplacian coordinates are linearly interpolated and the absolute coordinates are computed back. However, such a technique is suitable only for local morphing and the region of interest (ROI) has to be manually selected by user. The ROI boundary vertices acts as delimiter between the region which is transformed and the rest of the mesh which has to remain fix.

The key problem in approaches like [Kan00] or [Mic01] which partition the models in multiple correspondent patches relates to the user ability to make an efficient dissection in which each patch of one model corresponds exactly to one patch in the other 3D object. In order to reduce the amount of manual intervention required, Yu and Chuang [Yu03] propose a similar technique, with the difference that closed models are initially cut only into two patches by connecting four user-specified points which are in correspondence in the source and target objects. Additional feature points can be specified by user.

For each mesh patch, a base model is derived by applying a sequence of half-edge collapsing operations [Gar97]. The feature vertices selected by user are included in the decimated meshes. An initial embedding for the base mesh is constructed with the help of the mean value coordinates parameterization method proposed by Floater in [Flo03] and optimized using a stretch minimization scheme similar to that introduced in [San01].

Let us note that in this case, the feature correspondence problem must be solved separately. Thus, all feature points in correspondence are aligned using a foldover-free warping algorithm [Fuj98]. Having the initial embedding of the base mesh with the aligned feature

points and the refinement sequence, a coarse-to-fine parameterization is performed by remeshing the models uniformly or adaptively as in [Mic01].

In order to interpolate between different models, both linear and spline interpolation schemes, achieved in the spatial domain, are performed. Moreover, since the models are represented at different levels of resolution, the authors propose also an interpolation in the wavelet domain, which makes it possible to control interpolation starting time and speed at various resolutions (Figure III.22).

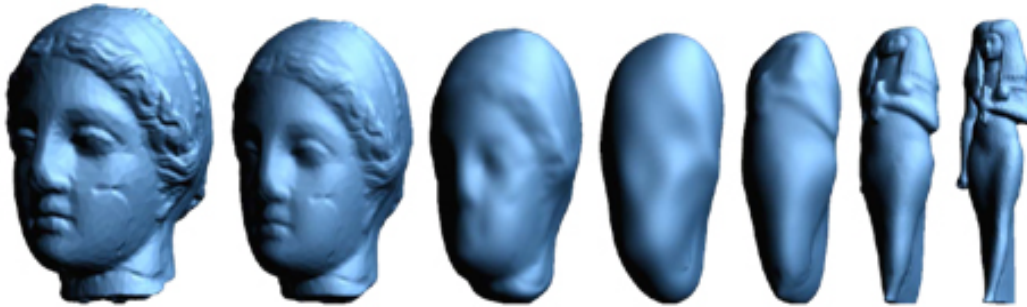


Figure III.22. Morphing of scheduled interpolation in wavelet domain [Yu03].

The computational time required for parameterization and remeshing of two objects with 10000 faces each is about 38 sec (35.69 for parameterization and 2.38 for remeshing) on a 1.5Ghz AMD Athlon XP PC [Yu03].

Let us note that an advantage of the methods based on multiresolution interpolation consists in the fact that multiple models can be morphed simultaneously, since multiple shapes can be approximated with the same set of vertices. An example of multi-target morphing is presented in Figure III.23.

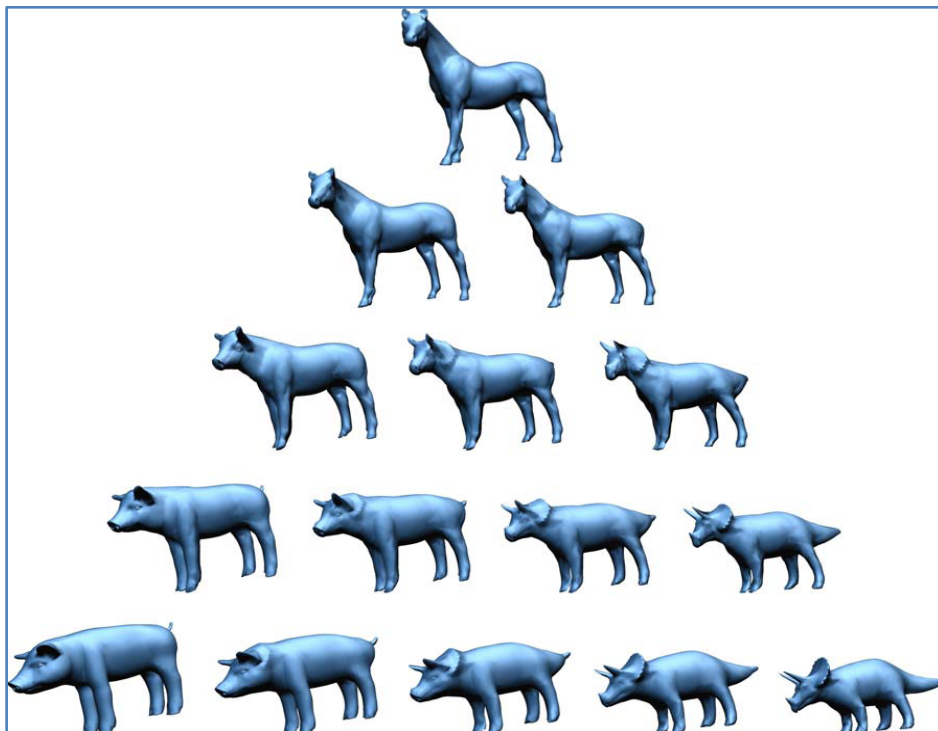


Figure III.23. Multi-target morphing [Yu03].

Lee and Huang [Lee03] propose the so-called SMCC (Structures of Minimal Contour Coverage) technique, which aims at speeding up the merging process (e.g. less than a second between two models of almost 4000 vertices compared with 8 seconds produced with the Alexa algorithm [Ale00]).

Given two 3D models, the user is required to select two sets of vertices on the models. The first set specifies the features needed to decompose the input meshes into several corresponding patches. The second one corresponds to additional control features, defined on each patch, for a finer correspondence control. Each patch is mapped into a n -sided regular 2D polygon. First, the n points associated to the patch and selected by user in the first stage are assigned to the n corners of the planar domain. Next, the other boundary vertices are mapped to an edge in the 2D domain. Finally, the interior vertices p_i are initially mapped to the center position $(0, 0)$, and iteratively displaced to a new position p'_i using the following relaxation equation:

$$p'_i = (1 - \lambda)p_i + \lambda \frac{\sum_{j=1}^{k_i} (w_{ij} p_j)}{\sum_{j=1}^{k_i} w_j} \quad (\text{III.5})$$

where k_i represents the number of adjacent p_j vertices at p_i . Parameter λ takes values between 0 and 1 and controls the movement speed. The weights w_{ij} help to preserve the aspect ratio of original triangles into the parametric domain.

Once the embeddings obtained, the control feature vertices have to be aligned. In order to solve this task, a free warping scheme based on a Gaussian radial basis function is employed. However, the warping can lead to foldovers. In this case, equation (III.5) is reiterated, while maintaining the feature vertices fixed.

A supermesh structure is constructed by overlaying and merging the models parameterizations. In contrast with other methods, like the one proposed by Alexa [Ale00] which assume that none of the vertices of one embedding lies on a vertex or an edge of the other graph, Lee and Huang [Lee03] identify and solve all intersection cases, including the degenerate ones. The algorithm starts with the source embedding and treats each target edge independently, by determining all its intersections and adding corresponding new vertices. Obviously, a nontriangular planar graph is produced and additional edges must be inserted to retriangulate it. The positions of the new vertices, relatively to the input models, are computed based on the barycentric coordinates. Once the metamesh is constructed, the morphing sequence is obtained by linear interpolation.

A very similar method with the one presented by Alexa in [Ale03] is proposed by Sheffer and Kraevoy in [She04]. The main difference consist in the fact that in contrast with Alexa's

method which make use of the Laplacian coordinates to interpolate between the models, in [She04] is introduced a new set of coordinates, so-called pyramid coordinates.

Pyramid coordinates measure the set of angles and lengths uniquely relating a vertex to its immediate neighbors (Figure III.24). Considering p a vertex in 3D, let p_1, p_2, \dots, p_m be its adjacent vertices. Given the normal $n(n_x, n_y, n_z)$ at p , the projection plane Π can be defined as:

$$\Pi = \{(x, y, z) \mid n_x x + n_y y + n_z z + d = 0\}, \quad d = \sum_{i=1}^m n \cdot p_i \quad (\text{III.6})$$

Denoting with p' and p_i' the projections of p and p_i to Π , the description of the vertex with respect to its neighbors can be expressed as:

- a set of angles α_i between the projected edges $\{p', p_i'\}$ and $\{p', p_{i+1}'\}$ (Figure III.24.b);
- a set of angles β_i between n and the edges $\{p', p_i'\}$ (Figure III.24.c);
- a set of projected edge lengths $l_i = \|p' - p_i'\|$.

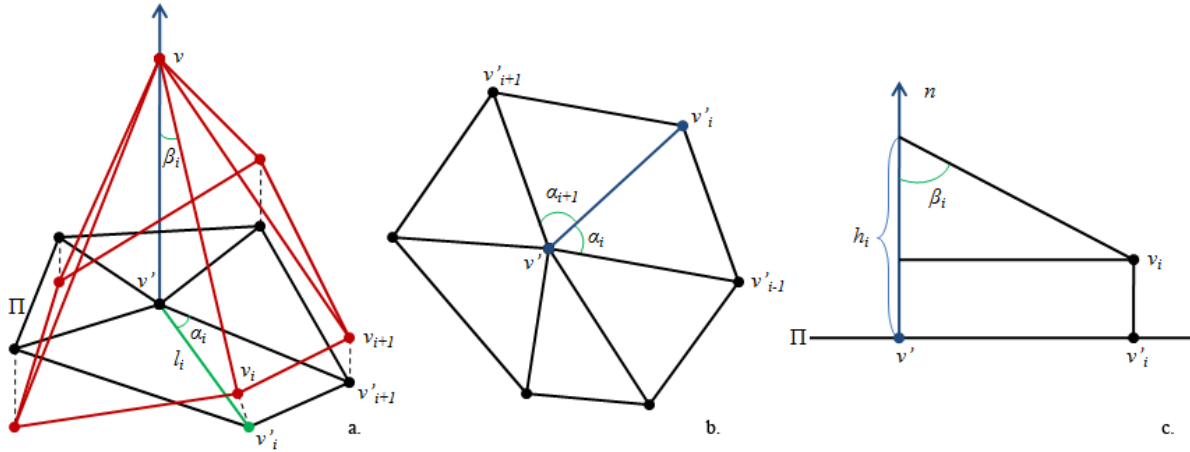


Figure III.24. (a) Pyramid coordinates; (b) tangential components in the projection plane Π ; (c) normal component β

Thus, pyramid coordinates represent a combination of tangential (α and l) and normal (β) components which have the property that both lengths and angles are invariant under rigid transformations.

In contrast with the Laplacian coordinates which are not invariant under rotation and scaling, the pyramid coordinates capture the local shape (lengths and angles) of the mesh around each vertex and help maintain this shape under various transformations.

Based on a small number of control vertices and a user-defined region of influence, the morphing procedure generates intermediate meshes which interpolate the shape properties of the input models. The method generates intermediate models based on interpolated pyramid coordinates. It can also take into account the trajectories of limited number of vertices provided by user. The user-specified vertices are linearly interpolated in time

between the source and the target values. For the remaining vertices, their positions are computed by a controlled reconstruction procedure using the pyramid coordinates.

As opposed to other existing methods, the proposed technique gradually transforms original objects eliminating the risk that same features disappear and then grow again. In addition, the technique is particularly well-suited for local morphing operations where just some parts of the model are modified.

An important advantage of this method is that although the algorithm does not explicitly prevent model self-intersections, the shape preservation property drastically reduces the risk of such self-intersections.

Starting from the approach of [Kan00], Urtasun *et al.* proposed in [Urt04] a fully automatic method. The two source and target meshes are aligned automatically and the feature vertices are determined without the user intervention using a modified version of the Iterative Closest Point algorithm (ICP) [Dew04].

The ICP algorithm orients the original meshes in the same manner and centers them at the same position. However, only a rough correspondence of source and target vertices is provided, particularly in the case where the two objects to be morphed present relatively different geometries. This can introduce disturbing artifacts in regions presenting salient features. Hopefully, such artifacts can be corrected using a local curvature matching.

As in [Kan00], the models are partitioned into patches and projected onto the plane using the harmonic map, with the only difference that in this case the boundary control vertices are established automatically.

The feature vertices are aligned using the warping method presented by Alexa [Ale00], adapted to the planar case. A supermesh is constructed with the help of a merging algorithm similar with the one proposed in [Kan00].

The interpolation between source and target geometries is then applied on the resulting supermesh. Here, a Slerp quaternion interpolation [Sho85] is preferred to the linear one, which makes it possible to obtain more smooth and realistic morphing sequences.

A different concept of mesh morphing is proposed by Ahn *et al.* [Ahn02], [Ahn04]. The approach aims to create a morphing sequence where, in addition to the geometric transformation, the mesh connectivity is also gradually changed. The main advantage of the method is related to the connectivity of the obtained in-between meshes, which is much simpler than the one of a supermesh.

Starting with the spherical embeddings proposed by Alexa [Ale00], with the feature vertices already aligned, the two mappings are overlaid in order to establish the vertices positions with respect to the 3D source (M^S) and target (M^T) shapes. Two in-between meshes (M'^S and M'^T) are constructed, each one containing a similar number of vertices (*i.e.*, $N_S + N_T - N_C$, where N_S and N_T are the number of vertices in source and target models while N_C is the number of coincident vertices obtained after the mappings are overlaid). The in-between mesh M'^S is constructed starting with the mapped source mesh topology to which the target vertices are added. If a vertex p^T of M^T can be mapped into a triangle f^S of M^S , then a new vertex p^S is added on the mapped position of p^T . The mesh is retriangulated by connecting the new vertex p^S to the three vertices of f^S . M'^T is constructed in a similar manner. Thus, M'^S and M'^T have an 1-to-1 correspondence, but M'^S has the same shape as M^S while M'^T has the same shape as M^T .

The next step consists of defining the transformation from the converted source mesh into the converted target. This task is accomplished through a mesh connectivity transformation procedure that employs a sequence of edge swap operations (Figure III.25). The order in which edges are swapped during the morphing is established based on a geometric error that takes into consideration the distance between a given edge and the edge created by the swap. Thus, during the morphing process, the connectivity is transformed gradually and the vertex positions are linearly interpolated.

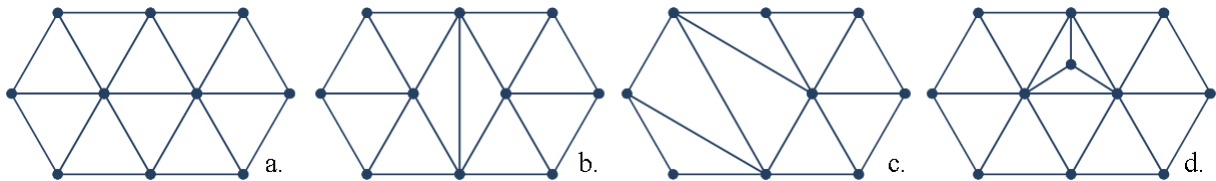


Figure III.25. Primitive operations used to transform the mesh connectivity: (a) ESO – edge swap operation; (b) VRO – vertex removal operation; VSO – vertex split operation.

Lin and Lee [Lin05a], propose a similar approach that aims to progressively transform the connectivity of the source model into that of the target during the morphing. The main difference consists in the fact that their algorithm avoid to creates the two in-between meshes M'^S and M'^T . A list of operations to be performed over the mesh vertices/edges is directly created on source and target embeddings using three primitives operations (Figure III.25): ESO (edge swap operation), VRO (vertex removal operation); VSO (vertex split operation).

The process of connectivity transformation is illustrated in Figure III.26. After the input models are mapped onto the parametric domain using a method proposed by the same authors in [Lee03], the two embeddings are overlapped. The almost incident vertices are merged and only the vertices of the source embedding are used to execute VRO and VSO

operations. Once a VSO is performed, a vertex is inserted on the source embedding. This operation might produce narrow triangles. Thus, local refinement is performed using ESO. Primitive operations are carried out iteratively until the source embedding is transformed into the target one. Finally, a priority control function is defined to establish an order in which the above operations will act during the morphing sequence.

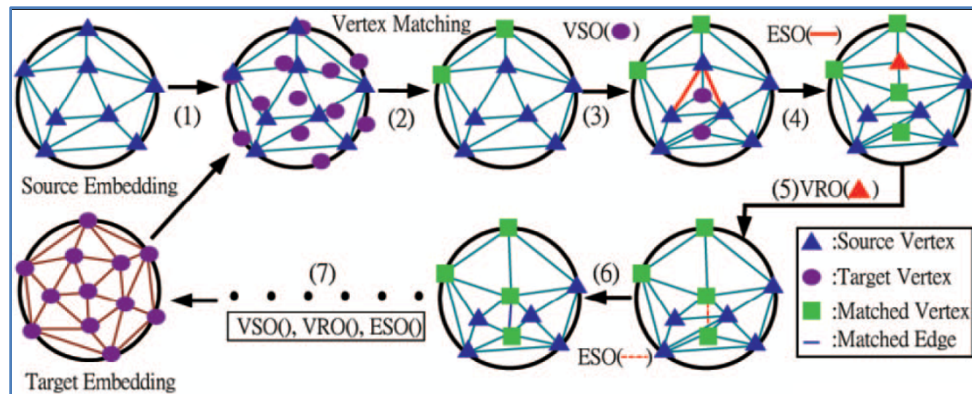


Figure III.26. The process of connectivity transformation employed in [Cha05].

This approach is further extended in [Lin05b] for the case of spherical embeddings. Specifically, there is a major difference required when an edge is inserted using a sequence of ESOs. On the spheres, an edge is an arc (*i.e.*, the shortest path) between two points. Therefore, the arc and its intersections with other arcs defined by other edges must be computed before the sequence of ESOs. In order to obtain the spherical parameterization the Alexa's algorithm [Ale00] is employed.

Kaneko *et al.* [Kan06] present a simple method to automatically establish a semantic topology match between two objects that have to be morphed. The purpose of their work is to set-up a model generation system that enables to create 3D shapes easily by morphing. This system first decomposes both original objects into several semantic elements, by considering the watershed method proposed in [Man99]. Here, a feature value need to be assigned to each vertex to generate a topographic map as a height function derived from the feature values. Usually a curvature measure associated to each vertex is utilized.

Then, the method automatically determines correspondences between each detected element/patch based on their relative location. Such regions are morphed one into each other independently using a simple algorithm which moves vertices from the initial surface toward the final one without modifying the vertex correspondence.

Even it is not directly related to mesh morphing, let us also mention the method proposed by Wu *et al.* [Wu07] due to its high potential for a possible morphing application. Their framework aims to give a solution for establishing a correspondence between arbitrary

meshes by directly mapping the connectivity of the source model onto the target mesh without needing to segment the input objects.

First, in order to establish a shape preserving correspondence between source and target meshes, a modified mean-value Laplacian fitting scheme is used. This operation is applied directly in the 3D space without being necessary to map the models onto a common parametric domain. The method achieves good results even with a reduced number of control features in the two models. In order to increase accuracy, a vertex relocation approach is proposed. Finally, each vertex is gradually projected onto the target model's surface to ensure a complete surface match.

Based on a spherical parameterization, Zhu and Pang [Zhu09] present a morphing algorithm for arbitrary genus-0 models which requires the user interaction in order to specify a set of feature pairs used to align the source and target meshes in the parametric domain.

The initial source and target models are first roughly aligned with the help of a principal component analysis (PCA). Then, the two PCA-normalized models are parameterized onto a common spherical domain. The spherical embedding is constructed using a relaxation operator $R(p_i)$ defined as follows:

$$R(p_i) = \frac{1}{d} \sum_{(i,j) \in \text{edges}} p_j, i \in \{1, 2, \dots, N\} \quad (\text{III.7})$$

where d is the degree of vertex p_i . Using the operator $R(p)$ in an iteratively manner for each vertex and then normalizing $R(p_i)$ to unit length in order to maintain all the vertices on the unit sphere, the mapping of M can be achieved.

The next step consists of refining the spherical parameterizations obtained such that each corresponding source and target feature points be placed to identical positions in the spherical domain. Starting from the spherical parameterization obtained the pairs of source and target feature points are first re-placed in their middle position on the unit sphere. The rest of the vertices from both source and target meshes are then iteratively re-distributed on the sphere with the help of the same relaxation operator $R(p_i)$. The authors claim that no triangle overlapping occurs even if a similar method is performed in [Ale00] and additional constraints are applied in order to avoid this problem. Based on the spherical embeddings, the positions of the source vertices relatively to the target shape are computed using barycentric coordinates. Finally, the morphing is obtained by interpolating the initial source vertices position with the new computed ones, without creating a metamesh. In this case, the target model could be only approximated with the source topology.

In [Ath12], Athanasiadis *et al.* present a method that performs morphing in a completely automatic manner, but which works well only in the case where similar source and target models, belonging to the same category are considered. Object alignment, feature detection and feature point matching is performed automatically with the help of a geometric local characteristic, so-called concavity intensity, inspired from [Sta07]. This feature is combined with an algorithm that detects the rapid variations of the surface normal, in order to obtain a region growing method that results in sets of points corresponding to the object individual features. The object features are then represented with the help of a connectivity graph that captures their adjacency information.

For each graph edge the geodesic distances between the centroid of the corresponding feature regions are computed. The graph is then simplified by collapsing edges that correspond to large distances. In addition, small regions that can introduce noise and are not significant are merged. The reduced adjacency graphs are used to perform a 3D alignment of the two models and establish a correspondence between the region patches.

For the initial mapping an improved Laplacian smoothing method is employed obtaining a spherical embedding which attempts to maintain uniform triangle areas and to avoid long edges. The Laplacian smoothing guarantees an unfolded mapping and preserves similarities with the initial mesh, but does not perform any triangle balancing and the mesh can degenerate. To avoid this, the authors use a weighted sum of the centroids of the surrounding triangles of each vertex, to determine their position.

In order to align the features of the target model with those of the source object, an objective function has to be minimized, under a set of geometric constraints. The alignment process is carried out on the spherical embedding. Thus, a first condition to be respected requires that each vertex $p_i(x_i, y_i, z_i)$ should lie on the surface of the unit sphere, as described in equation (III.14):

$$x_i^2 + y_i^2 + z_i^2 = 1, \forall p_i(x_i, y_i, z_i) \quad (\text{III.8})$$

In order to avoid triangle flipping, for each vertex of a face $f(p_0, p_1, p_2)$, it must be imposed that the vertex remains on the same side of the plane defined by the other two vertices and the center of the sphere:

$$(p_1 \times p_2) \cdot p_0 > 0 \quad (\text{III.9})$$

In addition to equation (III.15), the length of each edge must be preserved during the optimization:

$$p_i^S \cdot p_i^T = \|M(p_i^S) - M(p_i^T)\| \quad (\text{III.10})$$

where $M(p)$ is the initial position of vertex p on the original meshes. In this way, the morphology of the target object during the optimization process is preserved. This avoids introducing very long stretches of the mesh triangles.

The objective function to be minimized represents the sum of all inner products of every mapped feature vertex p^T of the target object with their corresponding feature vertex of the original mesh p^S :

$$\sum_{\forall p^T} p^T \cdot p^S \quad (\text{III.11})$$

The optimization described in equation (III.16) is solved with the help of the method introduced in [Wac06], which provides a nonlinear programming technique that handles problems with a large number of inequality constraints. However, the main limitation of the method is related to the computational bulk since the time for mapping an object with 5600 faces takes 15 minutes on an Intel Q6600 Core2 at 2.4GHz and GeForce 8600GT.

Once the spherical mapping of the two objects achieved, a merging process of the two topologies is performed. First, for each source edge, a list of intersections with the target topology is determined. Additionally, for each vertex, a list of the edges incident to it in clockwise order is calculated. Based on this data each closed bounded region is traversed in a clockwise order and the retriangulated merged topology is computed. Figure III.27 illustrates the merging process. Finally, the morphing sequence is performed based on a linear interpolation using directly the GPU (GLSL shaders).

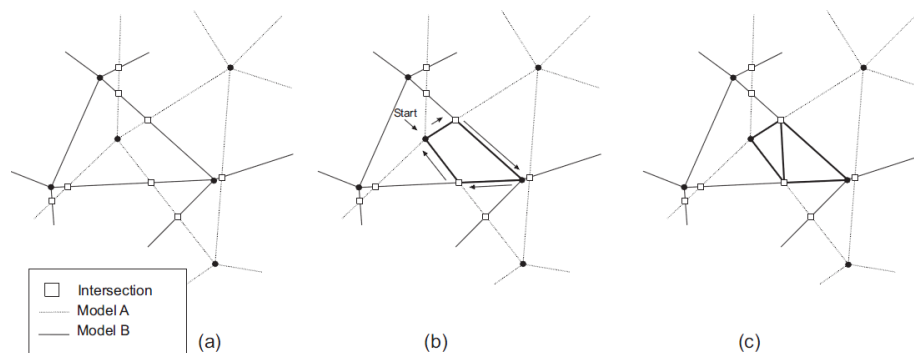


Figure III.27. The process of metamesh construction [Ath12].

The analysis of the state of the art shows that several phases involved in the morphing process are crucial for ensuring the quality of the resulting metamorphosis sequence. They notably concern:

- The parameterization method involved, which should guarantee low geometric distortions in terms of lengths, angles and areas,
- The warping of the source and target parametric domains, which should simultaneously guarantee a good match between corresponding feature points and a fold-over free deformation,

- The connectivity-related issues which should ensure a smooth and local adaptation between source and target models.

Based on these considerations, we propose a 3D mesh morphing framework described in the next section.

III.5. OVERVIEW OF THE PROPOSED 3D MESH MORPHING FRAMEWORK

The main steps involved in the proposed mesh morphing framework are illustrated in Figure III.28.

The morphing method includes the following steps:

1. **3D model normalization** - Since mesh models can be generated using a variety of techniques (e.g., 3D designers use CAD software, optical devices of 3D scanners) and, therefore, most of the 3D models available over the internet may have arbitrary scales, orientations and positions in the tridimensional virtual space, we first employ a PCA-based normalization [Joi02] in order to align the object with respect to its principal axes and scale it to the unit sphere.

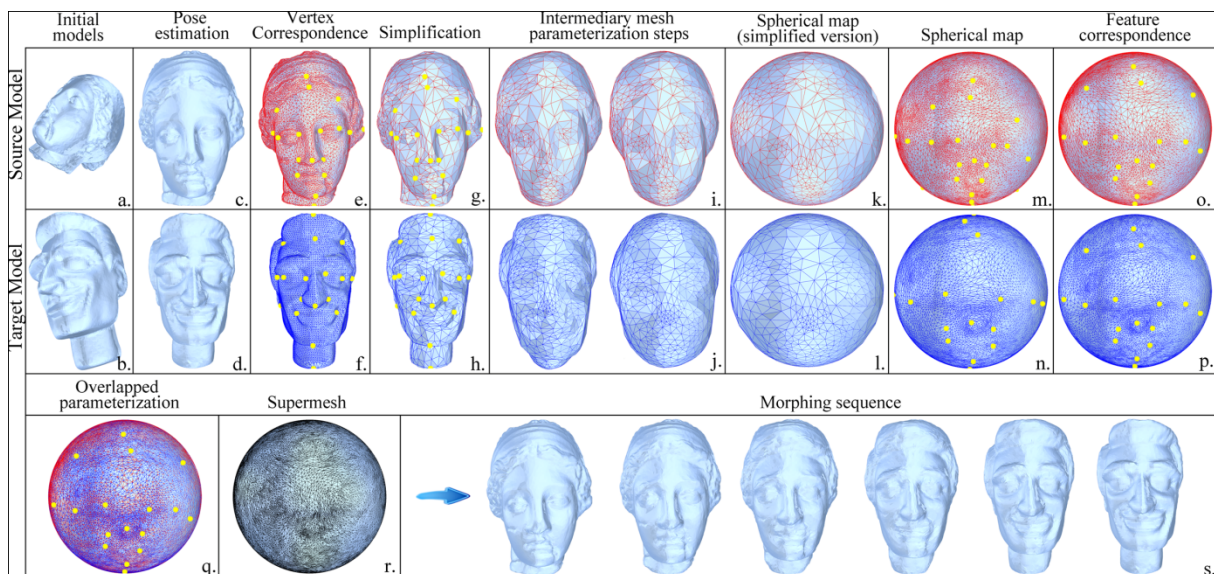


Figure III.28. Steps involved in our morphing process.

2. **Mesh simplification** – Since the parameterization process may require relatively important computational resources when dealing with highly complex meshes, described by thousands of vertices/triangles, we introduce a pre-processing step to produce coarser versions of the input meshes by iteratively reducing the number of vertices and triangles. In our work, we have adopted the QME surface simplification scheme introduced in [Gar97], and known as the QSlim method. A slight modification of the baseline technique has been

considered. Thus, in contrast with [Gar97], which can contract arbitrary two vertices, even if they are not connected by an edge, our method solely collapse adjacent vertices. This guarantees the preservation of the original model topology, which is strong constraint within the context of mesh morphing applications.

3. **Mesh parameterization** – Both the source and target models are mapped in a common parametric domain (*i.e.*, planar domain if the models are open or spherical domain if the models are closed) in order to establish the correspondence between the main features of the two objects. We introduce here two novel methods: one that concerns a planar parameterization technique, so-called edge length ratio preserving (ELRP) parameterization and another one dedicated to spherical parameterization based on a Gaussian curvature criterion.

4. **Feature correspondence** - In order to ensure that the main characteristics of the models are preserved during the morphing process, it is necessary to re-place the user specified corresponding feature points such that they share the same position in the parameter domain. Such a re-placement requires a global deformation of the whole parametric domain, such that the corresponding meshes should be smoothly deformed without foldovers. The process is referred to as mesh warping. In order to accomplish this task we make use of radial basis functions that allows to displace all mesh vertices based only on the known displacement of some control points (feature vertices).

5. **Pseudo metamesh construction** - Once the two input models are parameterized in a common domain and the main features of the objects are aligned properly, the next natural step in a morphing framework is to establish a one-to-one correspondence between the models shapes. In order to accomplish this task we introduce a simple yet efficient technique to create a pseudo supermesh which avoids tracking the edge intersections. In addition, our method reduces drastically the number of vertices normally needed in a supermesh structure.

6. **Interpolation** – The objective of the mesh interpolation step is to determine appropriate trajectories for each vertex connecting the initial position, defined on the source surface, to the final position, defined on the target shape. We solve this step in a simple way by adopting a linear interpolation scheme.

III.6. CONCLUSIONS

In this chapter, we first set the generic principles of visual object metamorphosis, starting with the case of 2D images. Then, we provided an overview of the main 3D mesh morphing

techniques proposed in the state of the art. The analysis of the state of the art reveals that, whatever the technique is used, the morphing process involves two different processes: the feature mapping and the interpolation problem.

The correspondence problem, although intensively analyzed, remains a difficult step. In order to establish the correspondence between the vertices of the input models a great majority of approaches exploit a parameterization of the source and the target meshes over a common parametric domain. The feature correspondence issue is then established in the parameter domain, which can be either spherical (for genus 0 closed meshes)[Ale00], [Zhu09], [Ath12] planar (generally for open surfaces) [Kan98], [Kan00], [Yu03] or object-dependent (as in the case of MAPS parameterizations)[Lee99], [Mic01].

However, there are other approaches [Ahn04], [Lin05a], [Lin05b] which are based neither on disk nor spherical parameterization. These methods usually avoid creating an intermediate mesh which contain both source and target geometry as in the case of the embedding merging and multiresolution remeshing approaches. These methods aim to create a morphing sequence where beside the geometric transformation, the mesh connectivity is also changed. The drawback here is related to the difficulty of interaction with the end user.

In the majority of cases, user intervention is required in order to specify some feature points and to establish the correspondence between the two objects. Some automatic solutions are proposed in the case of similar models, corresponding to a same category of objects [Urt04], [Ath10]. However, in the general case where morphing between arbitrary objects is required, such methods provide poor quality results.

Concerning the interpolation issue, the most frequently used approach is the linear interpolation, which offers the advantages of speed and simplicity. However, this simple method can cause self-intersection and shape degeneration which is usually not a very pleasant effect. An interpolation of higher degree or in other spaces (*i.e.*, Laplacian coordinates or Pyramid coordinates) is also possible, but without guaranteeing fold-over free morphing sequences in all cases.

Finally, we have introduced the proposed morphing framework, with the various stages involved, which includes normalization, mesh simplification, parameterization, warping, meta-mesh creation and interpolation.

IV. MESH PARAMETERIZATION

Summary: *This chapter introduces two main contributions of our work: The first one concern an enhanced 3D object planar parameterization method introducing a new barycentric mapping algorithm based on the length ratio preservation. A major advantage of our method, concerns the bijectivity property, which holds in all cases, and ensures valid and shape-preserving embeddings for arbitrary open and triangular 3D meshes, regardless their complexity. The second proposed approach represents a spherical parameterization method which exploits the Gaussian curvature associated to the mesh vertices. Valid spherical embeddings are obtained by locally flattening the mesh in an iterative manner, starting from vertices with maximal curvature values. This principle makes it possible to define a sequence of flattening operations that transform the initial mesh into a rounded, sphere-like surface that can be mapped onto the unit sphere.*

IV.1. INTRODUCTION

When considering a morphing process between two different source and target 3D meshes, the main difficulty to be overcome is related to topological problems related to different connectivities, numbers of vertices/faces that can describe the source and target shapes. Obviously, it is impossible to associate in a bilateral manner one vertex from the source to one vertex in the target model. The only solution is to consider the mapping between the two meshes as a mapping between their corresponding \mathbb{R}^3 surfaces. Thus, for each vertex on the source (resp. target) model, a correspondent point on the target (resp. source) model has to be identified, as illustrated in Figure IV.1.

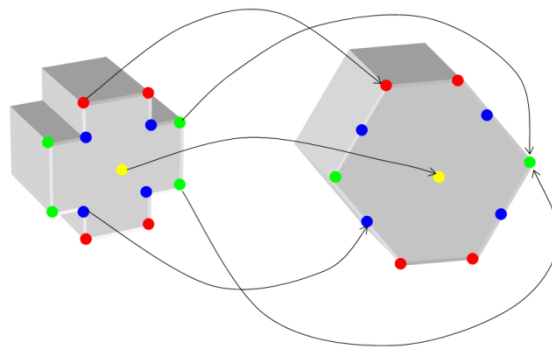


Figure IV.1. Vertex to vertex correspondences.

However, directly establishing such a correspondence is quite impossible, because of the complexity and diversity of shapes that can be modeled with 3D meshes. Instead, an indirect mapping method is preferred, which consists of:

1. Parameterizing both source and target models onto a common, parametric domain;
2. Warping the parametric domains in order to ensure a feature correspondence between the two 3D shapes to be morphed.

In this chapter, we will consider solely the first of the above-mentioned stages, which concerns the mesh parameterization.

In a general manner, the objective of any mesh parameterization method is to establish one-to-one mapping (bijective) mapping between the surface of a given 3D models and a given 2D parametric domain.

Parameterizing the surface of a 3D model digitally represented as a collection of flat polygons in \mathbb{R}^3 , was firstly used in the computer graphics field in order to map textures onto surfaces [Hak00]. More recently, parameterization became an essential phase in numerous mesh processing applications such as surface-fitting [Pie01], mesh-editing [Bie02], re-meshing [Smi06], compression [All05] and morphing [Zhu09].

The interest of 3D mesh parameterization techniques comes from the fact that complex operations that are intractable on the original 3D surface representation can be performed easily on a simple parametric domain such as the unit disk or the unit sphere. Various methods were developed for different kinds of parameter domains and parameterization properties. Before detailing them, let us first briefly establish some terminology.

The parameterization of a given 3D surface $S \subset \mathbb{R}^3$ is defined as a homeomorphism $\Phi: S \rightarrow D$ which maps the surface S over an appropriate 2D domain $D \subset \mathbb{R}^2$. In the case of 3D meshes, a parameterization is defined as a piece-wise linear embedding. More precisely, let $M = (V, E, F)$ be a 3D triangular mesh, where V , E and F respectively denote the sets of vertices, edges and triangles.

The parameterization of the mesh surface is completely specified by a function $\Phi: V \rightarrow D$, which associates to each vertex p_i of V a point $\varphi_i = \Phi(p_i)$ in the 2D domain D . This process is illustrated in Figure IV.2.

The bijection is required because each triangle of the mesh needs to have an appropriate image in the parameter domain. In other words, the faces in the parameter domain must not overlap.

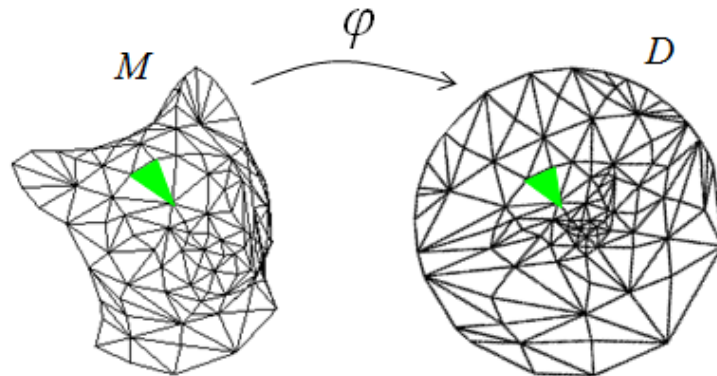


Figure IV.2. Mesh parameterization.

With the introduction of the parameterization paradigm, we also have to distinguish two spaces – the object domain, *i.e.* the space in which the mesh is defined (generally \mathbb{R}^3), and the parameter domain, *i.e.* the space in which the mesh is mapped onto.

The selection of an appropriate 2D parametric domain D depends in most of the cases of the original model topology. For open triangular meshes, the intuitive way to obtain a parameterization is to map its vertices in a planar domain. For closed, genus 0 models, a spherical domain (*i.e.*, the unit sphere) is more appropriate.

In practice, guaranteeing a valid parameterization (*i.e.*, continuous and bijective mapping function) is not straightforward. In particular, the phenomenon of triangle flipping (or mesh folding) can occur (Figure IV.3).

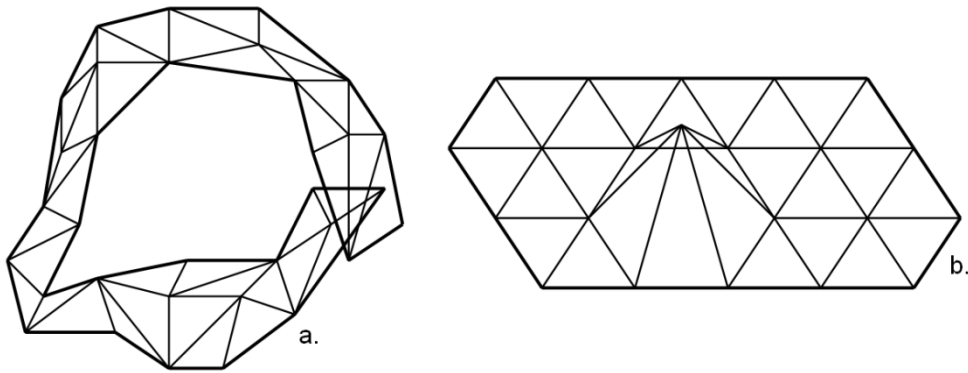


Figure IV.3. Fold-overs which lead to an invalid parameterization:
 (a) Boundary intersection; (b) Triangle flip.

Numerous approaches [Des02], [Flo03], [Fri05], [Sor02], [She02] have been introduced in order to prevent the triangles from flipping. There are different measures to avoid overlaps depending on the type of fold. In general there are two causes for such folding:

- The boundary of the parameter domain intersect itself (Figure IV.3.a) - This happens only in the case when the boundary is not predefined in the parametric domain, but can be handled by cutting along the borders of the intersection as described in [She01].
- Triangle flips (Figure IV.3.b) – This is the case when two adjacent triangles have opposite orientations. The mesh triangles orientation can be verified in the following manner: the vertices of the two adjacent triangles $f_m(p_i, p_j, p_k) \in F$ and $f_n(p_j, p_k, p_l) \in F$ are ordered in such a way that their associated coordinates in the parametric domain are in clockwise order; if we have the same order on their common edge $e_c(p_j, p_k)$ then we can say that one triangle lies on the other, *i.e.* their normals are flipped. Mathematically, we can check this by evaluating the following expression:

$$\text{sgn}((\varphi_i \times \varphi_j) \cdot \varphi_k) \quad (\text{IV.1})$$

where sgn is the signum function and $\varphi_i, \varphi_j, \varphi_k$ are positions of the vertices of a triangle in the parameter domain. Relation (IV.1) must be evaluated for each face. The parameterization is valid if all the triangles are oriented in the same way, *i.e.*, the signum of the result of the equation (IV.1) is the same for each face.

Even if the above conditions are satisfied, we cannot always speak about a “good” parameterization. Ideally, it is preferable that all triangles in the parametric domain have the areas proportional to those in the original space. However, in this case, the resulting triangles in the parametric domain risk to become degenerated / elongated and with disturbed aspect ratios.

On the contrary, if we try to preserve the angles in the parametric domain, the area distortion might significantly increase. As a result, the 3D surface details cannot be properly described under such a parameterization.

In this context, Floater and Hormann [Flo05] define three types of mappings:

- **Conformal mappings** – If the angles of any triangle in the parameter domain D are the same as those of the corresponding triangle in the original space M then that mapping is called conformal or angle preserving.
- **Equiareal mappings** – If the area of any triangle in the parameter domain D is the same as that of the corresponding triangle in the original space M then that mapping is called equiareal, authalic or area preserving.
- **Isometric mappings** – If the length of any edge in the parameter domain D is the same as that of the corresponding edge in the original space M then that mapping is called isometric or length preserving.

Let us note that it can be demonstrated that every isometric mapping is conformal and equiareal, and every conformal and equiareal mapping is isometric [Kre59].

In other words, isometric mapping is the ideal parameterization due to its zero distortions, which fully preserves angles and areas. Unfortunately, such an ideal parameterization can be determined solely in a small number of relatively simple cases. For example, when mapping into the plane, only developable surfaces (such as cylinders or cones) can admit planar isometric parameterization (Figure IV.4). For other, more general and complex surfaces, distortions must be tolerated, but minimized.

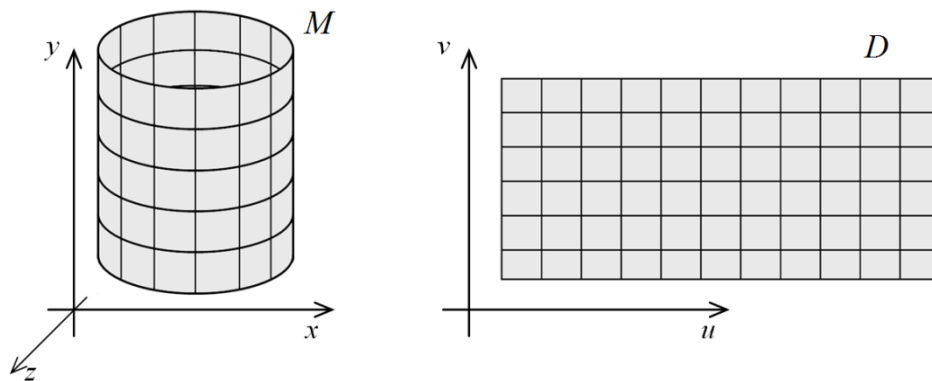


Figure IV.4. Isometric parameterization of a cylinder.

The majority of the approaches proposed in literature attempt to determine either a conformal mapping or an authalic mapping. In addition, they aim to minimizing some combination of angle and area distortions. The following sections will detail such methods for both planar and non-planar parameterizations.

IV.2. PLANAR PARAMETERIZATION OF TRIANGULAR MESHES

The first research works presented in the field of mesh parameterization concerned the planar mapping of meshes with disk-like topology. Such approaches are based on the principle that a mesh can be compared with a physical system where mesh edges are

springs that are connected to the vertices. If we consider fixed the mesh boundary in the parameter domain, then the inner vertices will relax in a configuration with minimum energy.

For illustrating this process, let us consider the 1D example presented in Figure IV.5. Considering a sequence of n points $p_1, p_2, \dots, p_n \in \mathbb{R}^2$ to be parameterized in an interval $[a, b] \subset \mathbb{R}$ and connecting each pair of consecutive points (p_i, p_{i+1}) with a spring, we obtain a chain of $n-1$ springs (Figure IV.5.a).

If all the points are forced to lie on an imaginary line with the endpoints p_1 and p_n fixed on a , respectively on b , then the result will be a contracted system with high energies stored in springs (Figure IV.5.b).

Releasing the endpoints of the system, the springs will relax freeing the energies (Figure IV.5.c). This principle also applies in the case of 3D meshes parameterizations over 2D domains. The difference here is that the springs energies cannot be completely eliminated. In this case, the objective is to relax the system in a configuration with minimum energy.

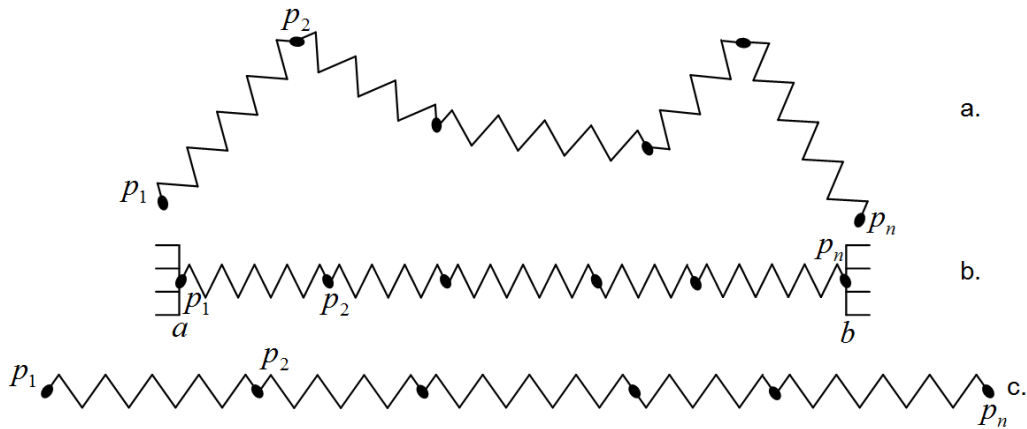


Figure IV.5. Parameterization of a spring model: (a) original spring system; (b) parameterization with fixed boundary; (c) system relaxation.

As a first planar parameterization method, let us mention the approach introduced by Eck *et al.* [Eck95], which consists of a generalization of the basic method proposed by Tutte [Tut63] for a planar graph. The spring energy is expressed in the following form:

$$E = \frac{1}{2} \sum_{\{i,j\} \in \text{Edges}} w_{i,j} \|\varphi_i - \varphi_j\|^2 = \frac{1}{2} \sum_{\{i,j\} \in \text{Edges}} w_{i,j} ((u_i - u_j)^2 + (v_i - v_j)^2) \quad (\text{IV.2})$$

where $w_{i,j}$ represent the spring constant defined for each edge $e\{i, j\}$, φ_i, φ_j are the vertex positions and u and v - the coordinates in the parameter domain.

In order to obtain the minimum of this energy it is required that the partial derivatives of E with respect to u_i and v_i to be zero for all interior vertices φ_i :

$$\frac{\partial}{\partial u_i} E = \frac{1}{2} \sum_{j \in \text{Neighbors}(i)} 2w_{i,j} (u_i - u_j) = 0 \quad (\text{IV.3})$$

$$\frac{\partial}{\partial v_i} E = \frac{1}{2} \sum_{j \in \text{Neighbors}(i)} 2w_{i,j} (v_i - v_j) = 0 \quad (\text{IV.4})$$

Again, the boundary vertices are considered fixed in the parameter domain and their corresponding positions are pre-calculated.

If we analyze equations (IV.3) and (IV.4), we can observe that every inner vertex can be expressed as a convex linear combination of its neighbors:

$$\sum_{j \in \text{Neighbors}(i)} w_{i,j} (u_i - u_j) = 0 \Rightarrow u_i = \sum_{j \in \text{Neighbors}(i)} \lambda_{i,j} u_j \quad (\text{IV.5})$$

$$\sum_{j \in \text{Neighbors}(i)} w_{i,j} (v_i - v_j) = 0 \Rightarrow v_i = \sum_{j \in \text{Neighbors}(i)} \lambda_{i,j} v_j \quad (\text{IV.6})$$

Where (u_i, v_j) are the 2D parametric coordinates of a vertex i , and $\lambda_{i,j}$ denotes the normalized spring weights for an edge $e\{i, j\}$ expressed as:

$$\lambda_{i,j} = w_{i,j} / \sum_{k \in \text{Neighbors}(i)} w_{i,k} \quad (\text{IV.7})$$

Let us observe that:

$$\sum_{j \in \text{Neighbors}(i)} \lambda_{i,j} = 1 \quad (\text{IV.8})$$

In equations (IV.5) and (IV.6), if we consider N the total number of points belonging to the mesh and n the number of inner vertices (non boundary points), then we can separate the interior and the boundary vertices in the sum in the following manner:

$$u_i - \sum_{\substack{j \in \text{Neighbors}(i) \\ j \leq n}} \lambda_{i,j} u_j = \sum_{\substack{j \in \text{Neighbors}(i) \\ j > n}} \lambda_{i,j} u_j \quad (\text{IV.9})$$

$$v_i - \sum_{\substack{j \in \text{Neighbors}(i) \\ j \leq n}} \lambda_{i,j} v_j = \sum_{\substack{j \in \text{Neighbors}(i) \\ j > n}} \lambda_{i,j} v_j \quad (\text{IV.10})$$

Here, without loss of generality, we consider that the boundary vertices are first indexed from 1 to n and that the interior vertices have corresponding indices $j > n$. The position of the boundary vertices is considered as fixed.

Writing the two above equations for any interior vertices we obtain two linear systems of equation to be solved. These two systems can be expressed in the following matrix forms:

$$A \cdot U = B_U \quad \text{and} \quad A \cdot V = B_V \quad (\text{IV.11})$$

where the unknown $U = [u_1, u_2, \dots, u_n]^T$ and $V = [v_1, v_2, \dots, v_n]^T$ are columns vectors corresponding to the u and v coordinates in the parameter domain D ; $B_u = [b_{u1}, b_{u2}, \dots, b_{un}]^T$ and $B_v = [b_{v1}, b_{v2}, \dots, b_{vn}]^T$ are columns vectors with coefficients:

$$b_{ui} = \sum_{\substack{j \in \text{Neighbors}(i) \\ j > n}} \lambda_{i,j} u_j \quad \text{and} \quad b_{vi} = \sum_{\substack{j \in \text{Neighbors}(i) \\ j > n}} \lambda_{i,j} v_j \quad (\text{IV.12})$$

$A = (a_{i,j})_{i,j \in 1 \dots n}$ - is a $n \times n$ matrix with elements:

$$a_{i,j} = \begin{cases} 1 & \text{if } i = j \\ -\lambda_{i,j} & \text{if } i \neq j \text{ and } j \in \text{Neighbors}(i) \\ 0 & \text{otherwise} \end{cases} \quad (\text{IV.13})$$

The existence and uniqueness of a solution for (IV.11) is equivalent to the non-singularity of the matrix A and is proven by Floater in [Flo97]. He also demonstrates that if the weights are positive and the matrix is symmetric, then the obtained parameterization exists and it is guaranteed to be bijective (*i.e.* there will be no overlapped triangles in the parameter domain). This theory is summarized by Gotsman *et al.* in [Got03] as the following theorem:

*Theorem IV.1: Given a planar 3-connected graph with a boundary fixed to a convex shape in \mathbb{R}^2 , the positions of the interior vertices form a planar triangulation (*i.e.*, none of the triangles overlap) if and only if each vertex position is some convex combination of its neighbor's positions.*

The above-presented principle holds for arbitrary boundary to which the border vertices can be mapped onto. However, the selection of an appropriate shape for the boundary vertices might have a relatively important impact on the parameterization results, as discussed in the next section.

IV.2.1. Selection of the boundary's shape

The convexity of the D domain boundary is a necessary condition in order to ensure that all the solutions of (IV.11) to belong to D . Thus, the problem of overlapping borders can be easily avoided, without any boundary optimization methods, if a convex shape is retained.

However, the choice of an appropriate convex polygon for the boundary may affect the quality and usefulness of the results. A polygon with vertices on a unit circle may be a good boundary shape because all the points will be further away from the middle and vertices may be spaced easily on this circle at distances proportional to the edge lengths between them.

Various approaches, particularly in the field of texture mapping methods (*e.g.* [San01], [Yos04], [Flo02a]) use as a boundary a square or a rectangle due to its similarity with a bitmap texture.

Whatever the type of the considered parametric domain, the main issue to be solved is to map the boundary vertices accordingly.

Floater [Flo97] maps the boundary on the unit square or circle using chord length parameterization, while Greiner and Hormann [Gre96] determine the plane that optimally fits all the boundary vertices (in the least square error sense), and then orthogonally project them onto this plane.

The requirements that the boundary should be fixed on a convex polygon may cause high distortion near the frontier. To overcome this drawback, various methods have been developed to allow free boundaries which treat both interior and border vertices in the same manner, in order to obtain simultaneously the boundary map and the mesh parameterization. Naturally, a higher similarity between the 2D and 3D boundary will lead to a smaller parameterization distortion.

Within this framework, one of the first methods proposed is the Lee *et. al.* approach [Lee02], which creates a virtual boundary somehow fixed but more “natural”. Afterwards, others methods have been developed which require fixing only a few vertices in the parametric domain [Lev02], [Des02], while more recent researches focus on establishing full free boundaries [Cao10], [Liu08], [Zha05].

In Figure IV.6, several methods with different boundary shapes are presented. Figure IV.6.b and Figure IV.6.c represent the 2D embedding of the 3D mesh, shown in Figure IV.6.a, in a parameter domain with a circle (respectively square) like boundary, while Figure IV.6.d presents the parameterization of the 3D mesh in a free boundary domain.

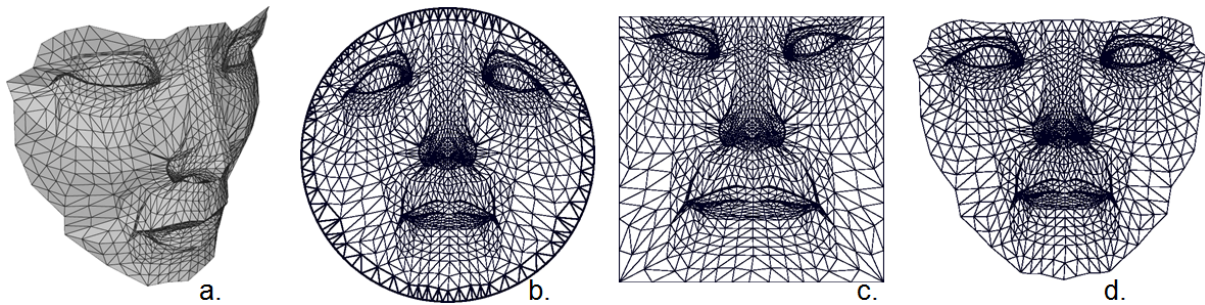


Figure IV.6. Parameterization with different bounding polygons[Lee02]: (a) 3D original mesh; (b) circle; (c) square; (d) free boundary.

In general, fixed boundary approaches offer the advantages of simplicity and of low computational complexity. In contrast, free boundary techniques generally produce significantly less distortion at the price of a higher computational cost.

Another important issue to be specified concern the definition of appropriate spring weights (equation IV.2) for guiding the parameterization process.

IV.2.2. Spring weights specification

In [Tut63], Tutte has chosen simple and uniform weights settings $w_{i,j} = 1$ if $\{i, j\}$ is an edge in the mesh. Tutte's objective was to compute straight line embeddings of planar graphs, within a theoretic setting. Later, his technique has been applied to texture mapping applications.

In such case, each point in the parameter domain is forced to be placed at the centroid of its neighbors. For this reason, the Tutte method has been called *barycentric mapping*.

In this context, equation (IV.7) can be rewritten as:

$$\lambda_{i,j} = 1/\deg(i) \quad (\text{IV.14})$$

where $\deg(i)$ is the degree or the valence of the vertex p_i .

Although the resulting mapping is proved to be bijective, a main drawback of this approach is that it doesn't fulfill the minimum requirement that would be expected from any parameterization method, which relates to the minimization of the geometric distortion measures. Thus, in practice the Tutte technique usually does not preserve any shape properties of the mesh because the choice of weights does not take into account any geometric property of the mesh, but solely its connectivity.

A significant amount of research has been dedicated to the optimization of the $\lambda_{i,j}$ coefficients, under the hypotheses of theorem IV.1.

The main objective is to minimize the different distortion components, such as angle deformation (harmonic/conformal parameterizations), length deformation, or area deformation (authalic parameterization).

The various approaches proposed are presented in the next section.

IV.2.2.1. Discrete harmonic map (DHP) and Discrete conformal map (DCP)

One of the first method proposed in this area is the so called *discrete harmonic map* introduced by Pinkall and Polthier [Pin93] in the context of differential geometry and adapted later by Eck et al [Eck95] for planar parameterization purposes. The goal of this approach is to minimize the Dirichlet energy, defined as:

$$E_{Dirichlet}(f) = \frac{1}{2} \int_M \|grad(f)\|^2, \quad (\text{IV.15})$$

where f is the mapping function. For a piecewise linear parameterization, corresponding to 3D meshes, equation (IV.15) can be reformulated, resulting in the following energy that has to be minimized:

$$E_{DCP} = \sum_{\text{Oriented Edges}\{i,j\}} \cot \alpha_{i,j} \|\varphi_i - \varphi_j\|^2 \quad , \quad (\text{IV.16})$$

with φ_i, φ_j - the vertex positions in the 2D parametric domain, and α_{ij} - the opposite left angle in the 3D space of the edge (i, j) (Figure IV.7).

Calculating the partial derivatives of E_{DCP} in respect with u and v - parametric coordinates – and imposing the necessary optimality conditions yields the following systems of equations:

$$\frac{\partial E_{DCP}}{\partial u_i} = \sum_{j \in \text{Neighbors}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(u_i - u_j) = 0 \quad (\text{IV.17})$$

$$\frac{\partial E_{DCP}}{\partial v_i} = \sum_{j \in \text{Neighbors}(i)} (\cot \alpha_{ij} + \cot \beta_{ij})(v_i - v_j) = 0 \quad . \quad (\text{IV.18})$$

The following weights are then obtained:

$$w_{i,j} = \cot \alpha_{ij} + \cot \beta_{ij} \quad (\text{IV.19})$$

where α_{ij} and β_{ij} are the opposite angles (in the 3D space) of the two triangles that share the same edge $\{i, j\}$ (Figure IV.7).

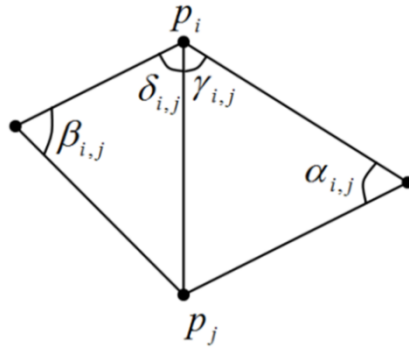


Figure IV.7. Angles used for weights computation.

As introduced, the discrete harmonic map aims to be an angle preserving technique. However since the boundary vertices need to be fixed in the parametric domain the resulting triangles near the frontier would be distorted in both areas and angles.

In order to overcome this limitation, Hormann and Greiner [Hor00] propose a free-boundary planar parameterization technique which requires that only two vertices to be fixed in the parametric domain. As in [Eck95], the mapping is determined as an energy minimization process which aims at maintaining low deformations. The energy to be minimized is defined as:

$$E_{\text{angle}}(F) = \frac{\cot(\alpha) \cdot |a|^2 + \cot(\beta) \cdot |b|^2 + \cot(\gamma) \cdot |c|^2}{2 \text{Area}(F_M)} \quad (\text{IV.20})$$

where a, b, c , are the edge length of triangle F that belong to original mesh, and α, β, γ are the angles in the parameter domain as presented in Figure IV.8.

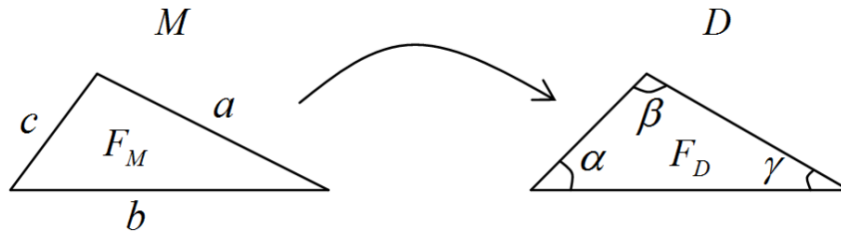


Figure IV.8. Edge and angle notation used in [Hor00].

The main advantage of this approach is that there is no more need to fix the parameter values of the boundary points in advance. Instead, the boundary of the parameterization will develop more naturally in such a way that the deformation energy is minimized.

Desbrun *et al.* [Des02] start by minimizing the Dirichlet energy obtaining the same weights as in (IV.19). In contrast with the baseline technique, they compute the boundary position as a part of the minimization procedure constructing a more natural free edge. Thereby, Desbrun is able to achieve a significant lower angle distortion, obtaining the so called *discrete conformal mapping*.

The harmonic and conformal mapping have the property to preserve the model shape, but not the area of the original mesh. Furthermore, the main drawback of these parameterizations is that the weights $w_{i,j}$ given by equation (IV.19) can be negative. According with the Floater demonstration [Flo97], this result can lead to non-bijective mapping and thus to triangle overlapping.

If we express equation (IV.19) in the following manner:

$$w_{i,j} = \cot \alpha_{ij} + \cot \beta_{ij} = \frac{\sin(\alpha_{ij} + \beta_{ij})}{\sin \alpha_{ij} \cdot \sin \beta_{ij}} \quad (\text{IV.21})$$

we can observe that the weights $w_{i,j}$ are positive if $\alpha_{ij} + \beta_{ij} \leq \pi$.

In practice, this constraint is rarely satisfied. A solution may consist of inserting additional vertices/edges in order to bisect the obtuse angles [Des02]. In a general manner, it is necessary to verify that the mesh topology satisfies the Delaunay triangulation [Del34] condition, which states that no mesh vertex should lie in the interior of the circumscribed sphere of any non-adjacent triangle.

Thus, Kharevych *et al.* [Kha06] demonstrate that if the mesh satisfies the Delaunay criterion, the parameterization obtained using the cotangent weights will always be bijective. In contrast with the baseline approach, Kharevych *et al.* formulate the discrete conformal

mapping in terms of circles and angles resulting from their intersection. The resulting method is called circle pattern parameterization.

The algorithm starts by assigning to each edge ($e_{i,j}$) of the mesh an angular weight $\theta(e_{i,j})$ which is expressed as:

- the intersection angle of the circumscribing circles of the incident triangles, in the case of an interior edge,
- the intersection angle of the circumscribing circle of the incident triangle with the considered edge, in the case of a boundary edge.

This can be summarized as described in the following equation:

$$\theta_e = \begin{cases} \pi - \alpha_{i,j} - \beta_{i,j} & \text{for interior edges} \\ \pi - \gamma_{i,k} & \text{for boundary edges} \end{cases}, \quad (\text{IV.22})$$

where $\alpha_{i,j}$, $\beta_{i,j}$ and $\gamma_{i,k}$ are edge opposite angles as illustrated in Figure IV.9. The dotted line represents here a boundary edge.

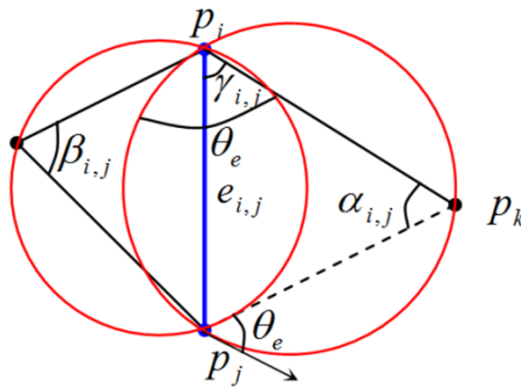


Figure IV.9. Angles used in the case of the circle patterns method.

These angles serve to incorporate the original geometry into the circle pattern technique. After the angles are assigned, a circle pattern is defined in the parametric domain, which is combinatorial equivalent to the initial triangulation constituting a so-called *coherent angle system* [Bob04]. A coherent angle system is by definition an assignment of angles for all triangles in plane which satisfy the following conditions:

- the angles are all positives;
- in each triangle the angles sum to π ;
- the angles satisfy the equation (IV.22), written in the parametric domain;

Based on the θ_e angles computed on the original model, a coherent angle system in the parametric domain is established by minimizing the following objective function:

$$E = \sum_k |\widehat{\theta}_{e_k} - \theta_{e_k}| \quad (\text{IV.23})$$

where $\widehat{\theta}_{e_k}$ is defined in the same manner as θ_{e_k} but in the parametric domain. This quadratic minimization problem, with the above presented constraints, is solved using the Mosek quadratic minimization technique [Mos05]. Based on these angles, the radius of the circles that define the mesh edges in the parametric domain, as well as the length of each edge is also determined through an energy minimization process.

Finally, the vertices coordinates u and v into the parameter domain are determined starting by placing one interior edge and then iteratively adding one edge after another by taking into account the previously computed angles and edge lengths.

The method offers the advantage of supporting meshes of arbitrary topologies. As a drawback, let us note that the resulting parameterization can contain overlaps. To overcome this problem, Kharevych [Kha06] introduced an optimization method based on cone singularity vertices (*i.e.*, vertices where angles of incident triangles do not sum to 2π) specified manually as boundary. Obviously, the inconvenient here is the amount of user interaction required.

A closely related approach is the angle-based flattening (ABF) algorithm of Sheffer and Sturler [She01]. Authors define free boundary parameterization in terms of angles specifying a set of constraints to be satisfied. The algorithm minimizes the relative deformation of the angles in the plane with respect to their corresponding angles in the 3D mesh. The objective function to be minimized is defined as:

$$F(\alpha) = \sum_{i=1}^{N_F} \sum_{j=1}^3 (\alpha_i^j - \phi_i^j)^2 w_i^j \quad (\text{IV.24})$$

where N_F represents the number of mesh triangles, α_i^j is the j angle ($j = 1, 2, 3$) on the i^{th} face in the 2D domain and ϕ_i^j is the *optimal angle* for α_i^j in the 2D parametric mesh. Here, w_i^j are positive weights defined as:

$$w_i^j = (\phi_i^j)^{-2} \quad (\text{IV.25})$$

The optimal angles ϕ_i^j are derived from the original mesh angles β_i^j as follows:

$$\phi_i^j(k) = \begin{cases} \beta_i^j(k) \frac{2\pi}{\sum_i \beta_i^j(k)}, & \text{if } p_k \text{ is an interior node} \\ \beta_i^j(k), & \text{if } p_k \text{ is a boundary node} \end{cases}, \quad (\text{IV.26})$$

where p_k is the mesh node to which the face f_i is attached to. In order to ensure a valid parameterization, the following set of constraints is imposed:

- 1) all mesh angles should be higher than zero;
- 2) the angles inside a triangle should sum to π ;
- 3) the angles around a point should sum to 2π ;

4) the following equation is satisfied for all interior mesh nodes:

$$\frac{\prod_i \sin(\alpha_i^{j(k)+1})}{\prod_i \sin(\alpha_i^{j(k)-1})} = 1 \quad (\text{IV.27})$$

Equation (IV.27) can be intuitively interpreted with the so-called wheel paradigm. If the set of all adjacent triangles to a vertex is considered as a wheel and all adjacent edges are seen as spokes, then the constraint (4) guarantees that after fixing the length of one (arbitrary) spoke, browsing over all spokes in counterclockwise order around the wheel, the length of the last spoke should be equal to the length of the first spoke. This constraint is illustrated in Figure IV.10 (where the constraint is violated).

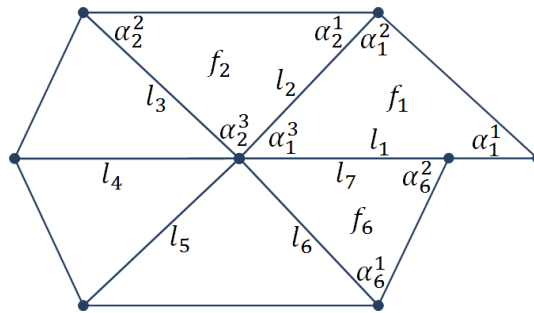


Figure IV.10. Incompatibility of edge length in a wheel paradigm [She01].

The constrained minimization problem is solved by employing a preconditioned iterative solver as proposed in [Van92].

The resulting map guarantees local bijectivity, but not a global one. It does not prevent the flat surface from generating self-intersections, in particular at the boundary level. To avoid this problem, additional constraints must be imposed and the algorithm reiterated.

Unfortunately, in practice, the ABF method proves to be slow (e.g., 158 sec for a model with 1032 triangles). In addition, in the case of meshes with a large number of vertices numerical stability problems appear. This is due to the iterative mechanism used to place the edges around a node which leads to error propagation. Each single vertex computation generates a small numerical error, but the accumulation of such errors for meshes with several thousands of triangles can be dramatic. Thus, in most of the cases the parameterization breaks out completely for models with more than 30K triangles.

In order to overcome such limitations, an improved technique, so-called ABF++ is introduced in [She05]. A new mechanism for computing the 2D angles is here proposed, which is based on sequential linearly constrained programming. This technique for solving constrained minimization problems considers the constraints as linear at each iteration. This simplifies the system at the expense of a slightly increased number of iterations.

In addition, in contrast to the baseline ABF method, the new technique formulates the conversion problem as a global linear system and computes all the vertex coordinates simultaneously. This avoids error accumulation. This minimization problem leads to the following system of equations that have to be solved in order to obtain the vertex position in the parametric domain:

$$\omega^f (\varphi_i - \varphi_j) + \varphi_k - \varphi_i = 0 \quad \text{for } \forall f = (i, j, k) \in F, \quad (\text{IV.28})$$

where:

$$\omega^f = \frac{\sin(\alpha_2^f)}{\sin(\alpha_3^f)} \cdot \begin{pmatrix} \cos(\alpha_1^f) & \sin(\alpha_1^f) \\ -\sin(\alpha_1^f) & \cos(\alpha_1^f) \end{pmatrix} \quad (\text{IV.29})$$

Here, α_1^f , α_2^f and α_3^f are the angles of a triangle f specified in counterclockwise order and φ_i , φ_j and φ_k their corresponding nodes in the parametric domain. For each triangle, two equations need to be written (for each of the u and v coordinates).

In order to eliminate the remaining degrees of freedom for the parameterization, four constraints are introduced by fixing two vertices that share a common edge. Thereby, a $2(N-2) \times 2(N-2)$ system of equations is created. Let us note that with respect to the initial ABF method, this leads to a speed-up in computation.

In order to reduce the drawback of the previous methods regarding the flipping triangles encountered due to the negative weights or the non-convexity of the parametric domain, Karni *et al.* [Kar05] propose an interesting method which consists of iteratively relocating the vertex position. The approach takes as input the results of an arbitrary mapping proposed in the above-presented methods as an initialization and then attempts to reduce the number of flipped triangles by reiterating the parameterization process. However, no guarantee of a flip-free final triangulation is proposed.

A well-known angle-preserving parameterization method is the *mean value coordinates* approach proposed by Floater in [Flo03]. Floater derives a generalization of barycentric coordinates, which allows a vertex in a planar triangulation to be expressed as a convex combination of its neighboring vertices. The approach is able to determine a new set of weights (IV.31) which has the property to be all positives keeping in the same time the simplicity of the Eck [Eck95] or Desbrun [Des02] approaches. The mean-value weights proposed are defined in the following manner:

$$w_{i,j} = \frac{\tan(\frac{\gamma_{i,j}}{2}) + \tan(\frac{\delta_{i,j}}{2})}{\|\varphi_i - \varphi_j\|} \quad (\text{IV.30})$$

where $\gamma_{i,j}$ and $\delta_{i,j}$ are the angles in the two triangles shared by the edge $\{i, j\}$ as illustrated in Figure IV.7. Based on these weights, similar matrix forms like that from equations (IV.9),

(IV.10) and (IV.11) can be constructed, with the difference that in this case the weights are not symmetric ($\lambda_{i,j} \neq \lambda_{j,i}$).

Although, the work of Tutte [Tut63] shows that in order to obtain a bijective mapping, the matrix has to be symmetric; Floater proves that mean-value parameterization is guaranteed to be bijective. However, in practice, there are some cases when the classical harmonic mapping preserves better the angles than the mean coordinates value approach.

Let us now analyze the area-preserving parameterization methods proposed.

IV.2.2.2. Discrete authalic map

The objective of *discrete authalic maps*, also called equiareal mappings, is to provide an area-preserving parameterization.

In [Flo05], Floater demonstrated that equiareal mappings, unlike the conformal ones, are not unique. Let us consider the example illustrated in Figure IV.11. Here, we can start from the left parameterization and construct different other mappings that preserve the areas (but not the angles). Thus, any attempt to minimize area deformation solely would lead to an ill-posed problem. For this reason, the majority of approaches combine the angular distortion minimization techniques with the ones of area-preservation [Pie10], [Dom10], [Des02], [Deg03], [Yan08].

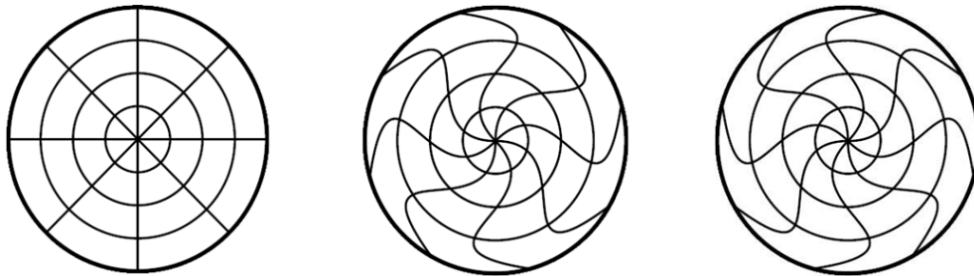


Figure IV.11. Equiareal mapping [Flo05]: In the three cases, the areas of the corresponding cells are identical.

Let us first mention the method introduced by Desbrun *et al.* in [Des02]. Here, authors introduce a tradeoff between angle and area distortions aiming to optimally (in the sense of an energy function to be minimized) map the 3D mesh onto the parametric domain. In addition to the conformal mapping approaches presented earlier, an area distortion metric is here included. The proposed energy function to be minimized is the Chi energy, defined by the following equation:

$$E_{\chi} = \frac{1}{2} \sum_{j \in \text{Neighbors}(i)} \frac{(\cot \gamma_{i,j} + \cot \delta_{i,j})}{\|p_i - p_j\|^2} (\varphi_i - \varphi_j)^2 \quad (\text{IV.31})$$

where the angles $\gamma_{i,j}$ and $\delta_{i,j}$ are defined as illustrated in Figure IV.7.

The critical point of the above energy could be determined by considering the set of partial derivatives, as described by the following relations:

$$\frac{\partial E_\chi}{\partial u_i} = \sum_{j \in \text{Neighbors}(i)} \frac{(\cot \gamma_{i,j} + \cot \delta_{i,j})}{\|p_i - p_j\|^2} (u_i - u_j) = 0 \quad (\text{IV.32})$$

$$\frac{\partial E_\chi}{\partial v_i} = \sum_{j \in \text{Neighbors}(i)} \frac{(\cot \gamma_{i,j} + \cot \delta_{i,j})}{\|p_i - p_j\|^2} (v_i - v_j) = 0 \quad (\text{IV.33})$$

A general distortion measure is then constructed as a linear combination of equations (IV.3), (IV.4) (IV.32), and (IV.33)

$$E_G = \mu E + (1 - \mu) E_\chi, \quad (\text{IV.34})$$

where μ is a real parameter taking values in the $[0, 1]$ interval that weights the conformal and authalic terms in the global energy.

If we take into consideration only the area energy ($\mu = 0$), we would obtain a full discrete authalic energy. However, the method measures deformations in the area distribution only locally within each one-ring of a considered vertex. In this way, the approach can accumulate a small error with each local deformation, resulting in an unbalanced global area distribution. The ratio between the initial area in 3D and the area in the parametric domain of a triangle is similar for adjacent faces, but may differ drastically with the one in other mesh regions.

Furthermore, it is not clear neither which is the optimal value for μ , nor if it depends on the considered 3D mesh models. Finally, let us point out that the method does not guarantee a valid planar embedding since the resulting linear systems obtained from (IV.32) and (IV.33) are not symmetric.

A closely-related formulation is proposed by Degener *et al.* in [Deg03]. Authors introduce a new method aiming to simultaneously preserve both angles and areas. The energies to be minimized, besides being invariant under rotation and translation of the domain, are also designed to prevent triangle flips and do not require a fixed boundary of the parameter domain.

The approach is actually an extension of a previous method proposed by Hormann and Greiner in [Hor00] which attempts to minimize only the angle distortion by optimizing a nonlinear functional that measures mesh conformability (equation IV.20). In addition, Degener *et al.* [Deg03] introduces an extra term that measures the area distortion of each triangle, defined as:

$$E_{area}(f) = \frac{Area(f_D)}{Area(f_M)} + \frac{Area(f_M)}{Area(f_D)}, \quad (IV.35)$$

where $Area(f_M)$ represents the f_M triangle area in the original mesh, while $Area(f_D)$ is the area of the same triangle in the parametric domain.

A real parameter θ allows the user to specify the relative importance of angle and area preservation in order to control the tradeoff between the related deformations; It is introduced in the following manner:

$$E_G(f) = E_{angle}(f) \cdot (E_{area}(f))^\theta \quad (IV.36)$$

The initial (non-optimized) 2D mapping is obtained through a hierarchical approach that computes progressive mesh sequences of the original mesh. The model is simplified until a base domain is obtained formed by an unique one triangle. The 2D coordinates of its vertices are initialized to a congruent triangle in the plane centered in the origin. Through vertex split operations all the removed vertices are iteratively reinserted into the mesh at the kernel center defined by its old adjacent nodes.

Based on the isometric distortion defined in equation (IV.37) for each face, an error E_i is computed for each vertex as the following partial sum:

$$E_i = \sum_{AdjacentFaces(i)} E_G(f) \quad (IV.37)$$

Then, a non-linear conjugate gradient optimizer [Pre94] is employed to establish the optimal position for each vertex in the parametric domain such that to minimize the E_i energy. Each vertex is treated separately while maintaining all other vertices fixed.

In addition to conformal and equiareal mappings, other researches are focused on preserving the relative distances across the mesh [Lee05], [Gre96], [Flo97]. Such approaches are presented in the following section.

IV.2.2.3. Distance preserving mapping

The idea of considering spring weights that are proportional to the lengths of the corresponding edges in the triangle mesh was first used by Greiner and Hormann in [Gre96].

They first orthogonally project the boundary vertices in a plane that best fits all these vertices in a least square sense. Then, the remaining points are forced to stay in the parameter plane in such a way that minimizes the following spring energy:

$$\sum_{\{i,j\} \in Edges} w_{i,j} (\|\varphi_i - \varphi_j\| - L_{i,j}^0)^2 \quad (IV.38)$$

where w_{ij} represents the weight for the edge e_{ij} that connects the original vertices p_i and p_j . This weight is defined as: $1/\|p_i - p_j\|^r$ for some real positive parameter r . Here, φ_i and φ_j are the corresponding positions in the 2D parametric domain of the corresponding vertices p_i and p_j , while L_{ij}^0 , defined as $\alpha\|p_i - p_j\|$ is the initial length of the edge e_{ij} , with α a real-valued and positive parameter.

In this framework, for $\alpha = 0$ and different values of parameter r we can encounter some well-known parameterizations. Thus, for $r = 0$, we obtain the uniform parameterization proposed by Tutte [Tut63]. In the case where $r = 0,5$ the centripetal mapping [Lee89] is achieved. Finally, for $r = 1$ we obtain the chord length method presented by Floater in [Flo97].

Let us also note that when parameter $\alpha = 0$, equation (IV.38) remains quadratic and positive defined in the unknowns w_{ij} . Its minimum can be consequently determined by solving a linear and sparse system of equations.

When parameter $\alpha \neq 0$ a better approximation of the model with a real spring system is achieved. However, the drawback here is that a non-linear optimization problem has to be solved.

Another approach which constructs the weights of the spring model based on the geodesic distances between points of the original 3D model is proposed in [Lee05]. Authors start from the idea proposed by Floater in [Flo97], but avoid computing areas on a complex surface. Instead, they exploit the barycentric coordinates in a triangle expressed as length ratios:

$$\begin{aligned} w_j &= |p_i p_{j'}| / (|p_j p_i| + |p_i p_{j'}|) \\ w_k &= (1 - w_j) |p_l p_{j'}| / |p_k p_l| \\ w_l &= (1 - w_j) |p_k p_{j'}| / |p_k p_l| \end{aligned} \quad (\text{IV.39})$$

where p_i, p_j, p_k, p_l are vertices represented as in Figure IV.12, and $p_{j'}$ is obtained so that the angles on each side of the line $p_j p_{j'}$ are equal. Once the weights are computed two linear systems of equation, similar to those in equation (IV.11) have to be solved in order to establish the 2D coordinates.

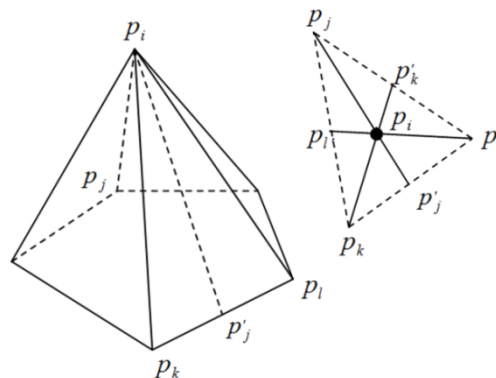


Figure IV.12. p_i vertex projection into the triangle p_j, p_k, p_l

More recently, in [Sun07] authors attempt to better preserve the shapes in the parametric domain with the help of straight distances computed with cutting planes. The principle is illustrated in Figure IV.13, where a base plane B is created locally for each interior vertex. The normal $Normal_B$ of the base plane B is computed by area-weighted averaging of neighboring face normals of p_i as:

$$Normal_B = \sum_{j \in Neighbors(i)} w_j Normal_j \quad (IV.40)$$

A cutting plane P passing through p_i, p_j and perpendicular on plane B is finally determined. The difference of the approach [Sun07] compared with the previous method [Lee05] consist in the way that the vertex p'_j is computed. Here, the point p'_j is obtained as the intersection of edge $\{p_k, p_l\}$ with the cutting plane P .

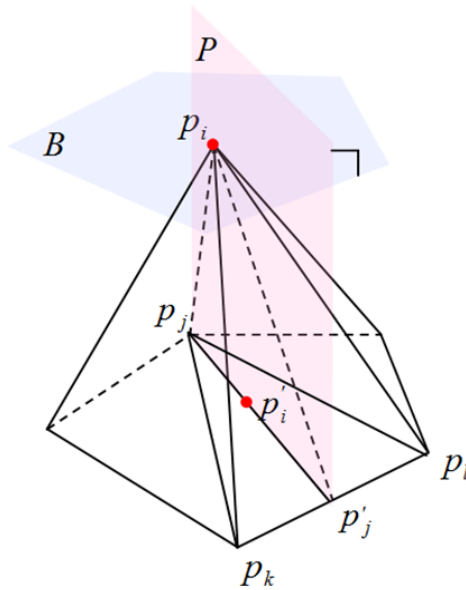


Figure IV.13. Local straightest path.

Table IV.1 summarizes the various planar parameterization techniques discusses in this section, with related principle, advantages and limitations.

Unfortunately, despite numerous existing planar parameterization techniques, as the analysis of the state of the art shows, only few of them can ensure a valid embedding with low distortions reported to the original 3D model shape.

In the following section we introduce a novel method of planar parameterization that belongs to the distance preserving mapping approaches. The proposed technique attempts to jointly minimize angle and area distortion based on edge length ratios. A major advantage of our method, concerns the bijectivity property, which holds in all cases, and ensures valid and shape-preserving embeddings for arbitrary open and triangular 3D meshes, regardless their complexity.

Table IV.1. Description of planar parameterization methods

Method	Type of distortion minimized	Complexity	Computational time	Comments
[Tut63] Tutte	None	+ (Linear)	+	- simple and uniform weights $w_{ij} = 1$ - bijective mapping - do not preserve the shape of the mesh - fixed and convex boundary
[Eck95] Eck	Angles	+ (Linear)	+	- the parameterization may not be always bijective - require fixed and convex boundary - discrete harmonic mapping - do not preserve the area of the model
[Flo97] Floater	Angles	++ (Linear)	+	- shape preserving mapping - bijective parameterization - require a fixed and convex boundary - high distortion across the border
[Flo03] Floater	Angles	+ (Linear)	+	- bijective mapping - require a fixed and convex or star shaped boundary - high distortion across the border
[Des02] Desbrun	Angles and Areas	++ (Linear)	++	- almost free boundary (require to fix at least two border vertices) - the result is sensitive to the choice of the fixed vertices - possible triangles over lapping (non-bijective mapping) - suffer from high shrinkage
[Hor00] Hormann	Angles	+++ (Non Linear)	++++	- free boundary - bijective mappings - low distortions
[Deg03] Degener	Angles and Areas	+++ (Non Linear)	+++++	- the importance between angle and area preservation can be controlled manually - free boundary - use a hierarchical solver to speed up the nonlinear optimization - bijective mappings
[San01] Sander	Lengths	++++ (Non Linear)	+++++	- free boundary - create the parameterization using a coarse-to-fine optimization strategy - partition mesh into charts and map each one on the plane - applicable also for closed meshes
[Kha06] Kharevych	Angles	+++ (Non Linear)	++++	- free boundaries or user controlled boundary shape via prescribed curvatures - map meshes with arbitrary topology to the plane - meshes of genus zero can be parameterized over the sphere - the parameterization can contain global overlaps
[She01] Sheffer	Angles	++++ (Non Linear)	+++++	- local bijectivity, but not global (self intersection of the boundary; if this happens a number of optimization has to be made) - very slow - instable for meshes with medium number of vertices (>10k), and impractical for meshes with more than 30k vertices
[She05] Sheffer	Angles	+++ (Non Linear)	+++	- improve the [She01] method introducing a new numerical solution technique to speed up the parameterization - free boundary - can process models with millions of triangles relatively fast implementing a coarse to fine parameterization
[Sun07] Sungyeol	Lengths	++ (Linear)	+++	- free boundary - comparative distortions with the Floater method [Flo03] - there is no guaranty for a bijective mapping

IV.2.3. Edge length ratio preserving (ELRP) planar parameterization

The basic principle of the proposed method consists of defining a new set of weights, determined based on the local geometry of the original model.

Concerning the boundary specification issue, we have adopted a fixed and unique boundary, which is the unit circle. Let us note that this choice is without any loss of generality, since the proposed method can be adapted to any type of boundary (including free boundaries).

We first set an arbitrary border vertex on the unit circle and then place the rest of the vertices along the boundary domain such that the geodesic distances between them to be proportional to the original edge lengths. For this purpose, we let p_1, p_2, \dots, p_m be the ordered boundary vertices and identify $p_{m+1} = p_1$. The bijectivity of the parameterization is assured if the corresponding parameter points $\varphi_1, \varphi_2, \dots, \varphi_m$ of the boundary vertices form a convex polygon, which can be achieved by placing them on the unit circle. If we write the parameter points as:

$$\varphi_i = \begin{pmatrix} \cos \alpha_i \\ \sin \alpha_i \end{pmatrix}, \quad i = 1, \dots, m+1 \quad (\text{IV.41})$$

A reasonable measure of the distance between φ_i and φ_{i+1} is the corresponding arc length difference $\alpha_{i+1} - \alpha_i$ as the length of the arc between those two points (Figure IV.14). Thus, the parameterization of the boundary can then be regarded as a univariate mapping problem with parameter points α_i and fixed endpoints $\alpha_1 = 0$ and $\alpha_{m+1} = 2\pi$. The problem can be solved by writing:

$$\alpha_{i+1} = \alpha_i + \frac{2\pi}{d_i \sum_{j=1}^m 1/d_j} \quad (\text{IV.42})$$

where d_j represents the chord length weights defined as:

$$d_i = \frac{1}{\|p_{i+1} - p_i\|} \quad (\text{IV.43})$$

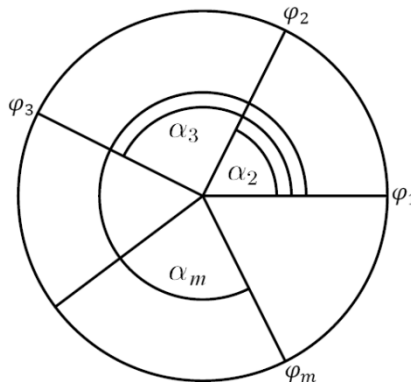


Figure IV.14. Parameterizing the boundary over the unit circle.

Establishing the weights for the spring system described by equation (IV.11) is highly important and can significantly affect the parameterization quality in terms of angle and area distortion. Most of the methods presented in the previous section returns less satisfactory results when considering the area distortion criterion. In order to overcome this drawback, we propose a new set of weights that minimize the areal deformation while maintaining a low angle distortion.

Each inner vertex of a mesh can be expressed as a linear combination of its neighbors. A weight, associated to each edge in the mesh (Figure IV.15), is computed as the ratio of the distance between the current vertex p_i and the adjacent vertex p_j normalized to the total sum of lengths for all edges incident to p_i , as described by the following equation:

$$w_{ij} = \frac{l_{ij}}{\sum_{j \in \text{Neighbors}(i)} l_{ij}} \quad (\text{IV.44})$$

When considering the above-defined weights, equation (IV.9) and (IV.10) can be rewritten in the following form:

$$u_i - \frac{\sum_{j \in \text{Neighbors}(i)} l_{i,j} u_j}{\sum_{j \in \text{Neighbors}(i)} l_{ij}} = 0 \Rightarrow -u_i \sum_{j \in \text{Neighbors}(i)} l_{ij} + \sum_{j \in \text{Neighbors}(i)} l_{i,j} u_j = 0 \quad (\text{IV.45})$$

$$v_i - \frac{\sum_{j \in \text{Neighbors}(i)} l_{i,j} v_j}{\sum_{j \in \text{Neighbors}(i)} l_{ij}} = 0 \Rightarrow -v_i \sum_{j \in \text{Neighbors}(i)} l_{ij} + \sum_{j \in \text{Neighbors}(i)} l_{i,j} v_j = 0 \quad (\text{IV.46})$$

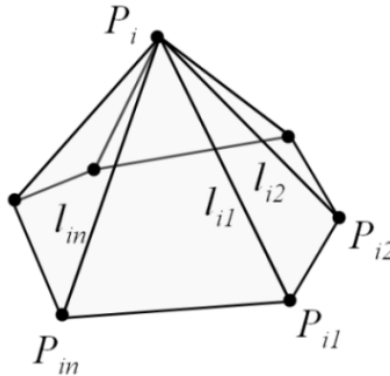


Figure IV.15. The one-ring neighbors of vertex p_i and the associated lengths.

As it can be observed, in this case the resulting system of equations is symmetric and all the elements from the matrix A (as defined in equation (IV.11)), excepting the main diagonal are positive. This property guarantees the bijectivity of our parameterization. Furthermore, the resulted matrix is sparse since the non-zero elements depend only on the adjacent vertices. This allows us to compute the spring system solution by using the conjugate gradient method [Pre02] that iteratively solves the sparse linear system.

IV.2.4. Objective experimental evaluation of planar mesh parameterization methods

In order to evaluate and validate the proposed ELRP parameterization method, we retained for comparison the following three state of the art approaches:

- *Uniform parameterization, which corresponds to the baseline planar graph method proposed by Tutte [Tut63].*
- *Mean Value Coordinates [Flo03], which aims to preserve the angles of the original mesh in the parametric domain.*
- *Harmonic Mapping – conceptually elaborated by Pinkall and Polthier [Pin93] in the differential geometry context, and integrated by Eck *et al.* [Eck95] for mesh parameterization purposes. The harmonic mapping aims to preserve the angles, but the considered weights not always ensure a valid parameterization.*

For each technique, we analyzed the mesh deformation measures in the parametric domain, in terms of angles, areas and lengths. More precisely, as evaluation metrics we have considered are the angle, area (surface) and length distortions (respectively denoted by D_A , D_S and D_L) as introduced in [Lee05] and defined as follows:

$$D_A = \sum_i^{3T} \left(\frac{\alpha_{iM} - \alpha_{iS}}{\alpha_{iS}} \right)^2 \quad (IV.47)$$

$$D_S = \sum_{i=1}^T \left(\frac{A_{i,M}}{A_{T,M}} - \frac{A_{i,S}}{A_{T,S}} \right)^2 \quad (IV.48)$$

$$D_L = \sum_{i=1}^N \sum_{j \in \text{Neighbors}(i)} \left(\frac{l_{ij,M}}{\sum_{j \in \text{Neighbors}(i)} l_{ij,M}} - \frac{l_{ij,S}}{\sum_{j \in \text{Neighbors}(i)} l_{ij,S}} \right) \quad (IV.49)$$

where T is the number of triangles, N represent the number of vertices, α denote the mesh angles and A represents the triangle areas. Indices M and S respectively indicate original and parameterized models. Ideally, all the three types of distortions should be as close as possible to zero.

We have considered an object corpus of 10 3D mesh models from the Princeton Shape Benchmark (<http://shape.cs.princeton.edu/benchmark/>) and from the MPEG 7 3D model test set (<http://3d.csie.ntu.edu.tw/>). The selected objects are open manifold triangular mesh models characterized by complex geometries and including various types of shapes.

Figure IV.16 and Figure IV.17 presents some visual results obtained after applying all the considered algorithms on some of our test models. As is can be observed, our method

always returns valid embedding for any arbitrary open and triangular 3D meshes, regardless their complexity.

The results synthesized in Table IV.2 present the distortions obtained with our proposed ELRP method, together with those corresponding to the state of the art. For each model and for each distortion criterion the best performances are marked in bold.

Table IV.2. Comparative study concerning area, angles and length distortions.

Model	No. vert	Uniform parameterization [Tut63]			Mean Value Coordinates [Flo03]			Harmonic Mapping [Eck95]			ELRP		
		D _S	D _A	D _L	D _S	D _A	D _L	D _S	D _A	D _L	D _S	D _A	D _L
Cow	1023	13.169	0.297	14.081	21.612	0.106	16.948	167.85	0.037	40.011	0.079	0.287	1.402
Chess horse	143	49.724	0.483	67.425	46.319	0.183	45.889	9.229	0.209	15.678	0.801	0.618	7.774
Lion	575	1.945	0.259	9.022	10.758	0.150	18.036	34.501	0.071	29.822	0.066	0.303	1.549
Delphin	355	0.795	0.331	3.755	409.06	7.910	7.885	Overlapping triangles			0.033	0.392	0.787
Cat	352	0.612	0.160	3.994	0.757	0.052	3.992	1.042	0.029	4.807	0.042	0.185	0.879
Hand	300	0.286	0.649	5.936	18579.1	0.587	608.36	Overlapping triangles			0.025	0.773	1.708
Statue	458	0.002	0.370	0.207	0.003	0.221	0.169	0.004	0.246	0.205	0.001	0.294	0.101
Face	1500	0.263	0.233	1.293	0.044	0.030	0.394	0.083	0.025	0.494	0.011	0.171	0.304
Beethoven	1200	0.003	0.283	0.179	0.001	0.083	0.071	0.001	0.072	0.065	0.001	0.219	0.078
Cat Head	135	0.177	0.159	2.251	0.104	0.057	1.352	0.098	0.043	1.311	0.027	0.175	0.567

Concerning the Tutte [Tut63] method, although the resulting mapping is bijective, the numerical examples show that this technique does not preserve any shape properties of the mesh. One reason for this bad behavior is that the choice of weights does not take into account the geometry of the mesh, but solely its connectivity.

The harmonic mapping globally preserves the model shape, but the corresponding areas are severely distorted. In addition, for some models the associated weights take negative values which leads to non-bijection and thus non-valid parameterizations. In the case of mean value coordinates even though the resulted matrix loses the symmetric property, the resulted embedding is valid in all cases. However, the major drawback of this method is related to the computational complexity because in this situation it is impossible to use the fast conjugate gradient algorithm to solve the linear systems involved. The analysis of the results obviously shows that the proposed length ration method outperform the other approaches in the case of both area (with a mean 78,5 % reduction) and length (with a global average of 57% reduction) distortions. For the angle distortion, the best performances are achieved by the harmonic mapping technique. However, the harmonic mapping fails in the case of some models due to the negative weights in the energy spring system. Thus, the proposed ELRP method offers the advantage of a larger applicability.

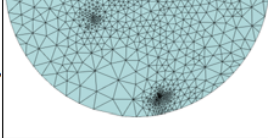
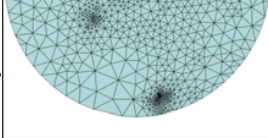
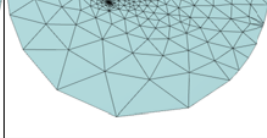
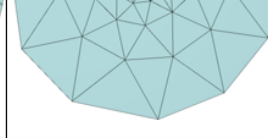
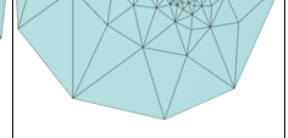
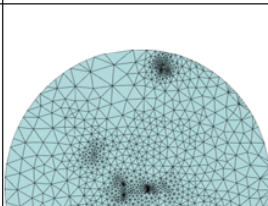
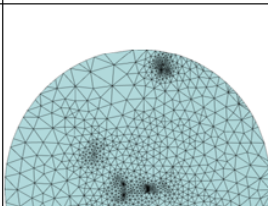
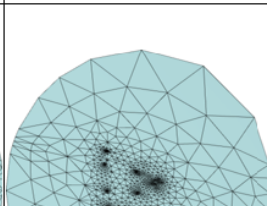
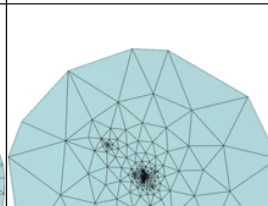
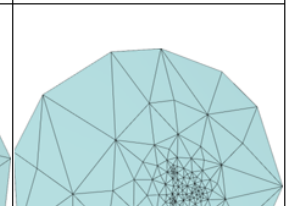
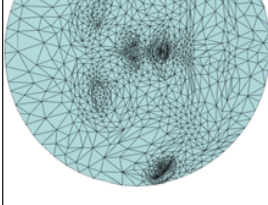
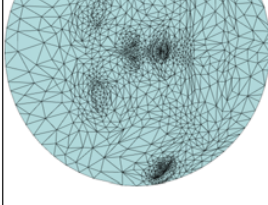
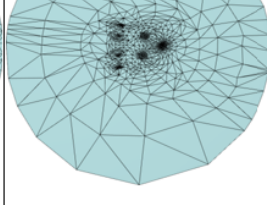
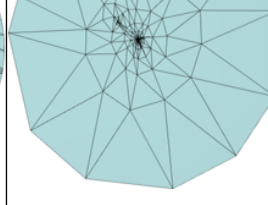
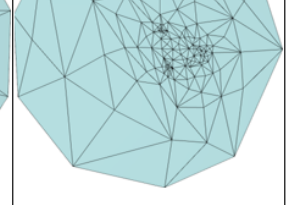
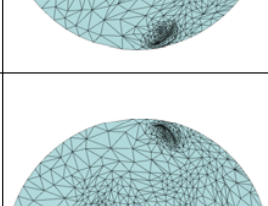
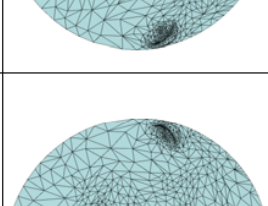
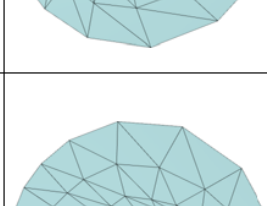
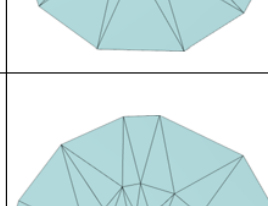
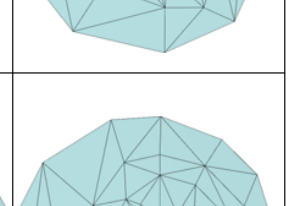
Model	Uniform parameterization	Mean Value Coordinates	Harmonic Mapping	Our length ratio preserving approach
				
				
				
				

Figure IV.16. Comparative visual evaluation of 3D mesh planar parameterization (1).

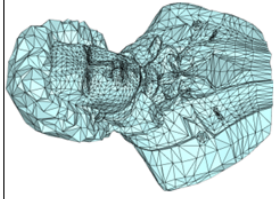
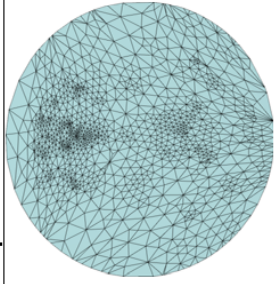
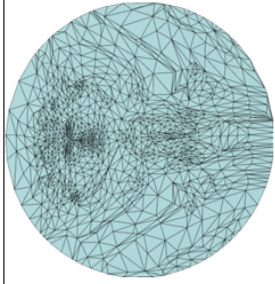
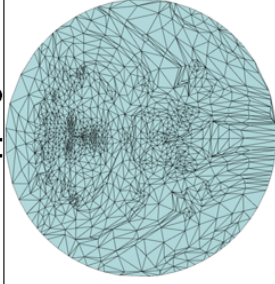
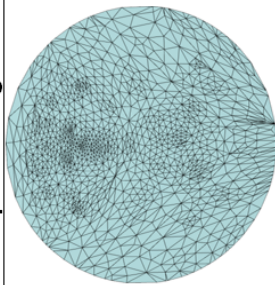
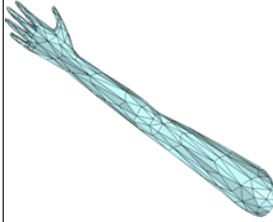
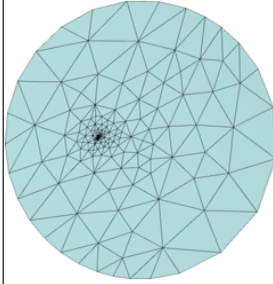
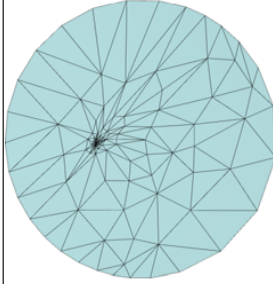
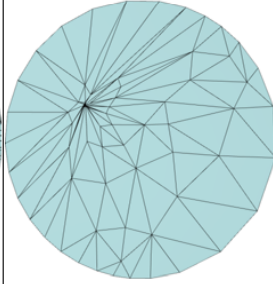
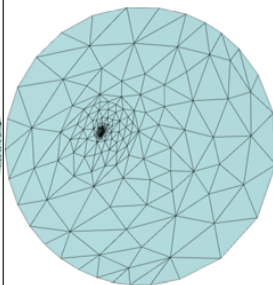
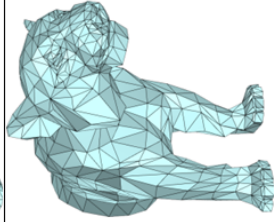
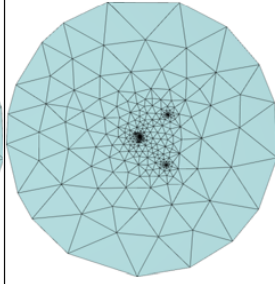
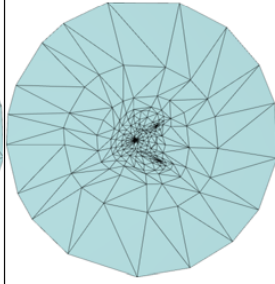
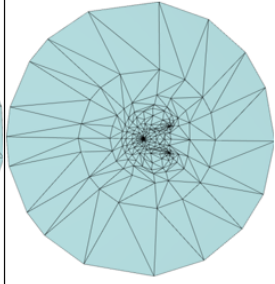
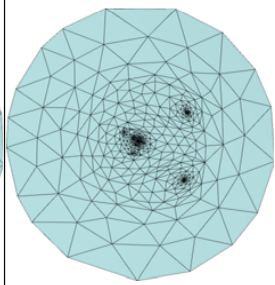
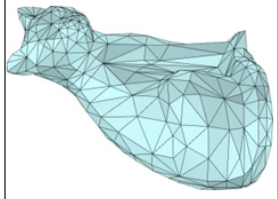
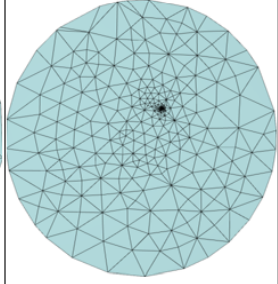
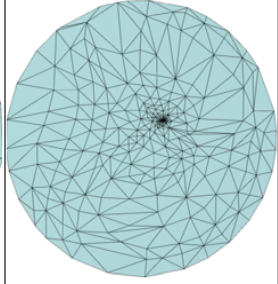
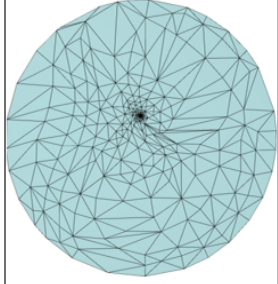
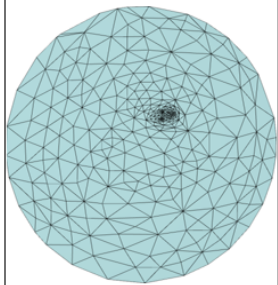
Model	Uniform parameterization	Mean Value Coordinates	Harmonic Mapping	Length ratio preserving
				
				
				
				

Figure IV.17. Comparative visual evaluation of 3D mesh planar parameterization (2).

IV.3. SPHERICAL PARAMETERIZATION OF TRIANGULAR MESHES

If for open, single-connected triangle meshes (*i.e.* disk topologically equivalent to the unit disk) the planar parameterization are naturally adapted, in the case of closed 3D meshes different solutions have to be investigated.

IV.3.1. State of the art on spherical embedding

The most straightforward method to handle closed 3D models is to create an artificial boundary by determining a closed path along the mesh edges and cut the mesh along the path. This process will result in two open patches that can be individually parameterized with respect to the unit disk by applying arbitrary planar parameterization methods. Different techniques of the literature adopt this paradigm [Pip00], [Sor02], [She02a], [She02b].

The simplest way to obtain the boundary which makes the object to be open is to eliminate an arbitrary triangle from the mesh [Cla04].

More sophisticated methods attempt to optimize the considered boundaries (Figure IV.18). However, even so, because of discontinuities introduced by the mesh cuts at the level of the boundary edges, the resulting distortions can be very high.

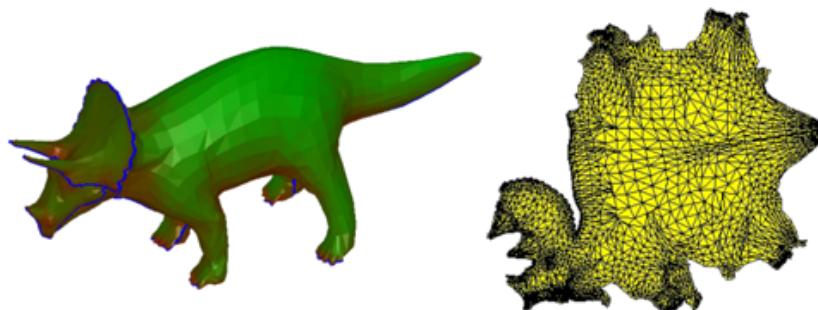


Figure IV.18. Planar parameterization of a closed genus-0 3D mesh by cuts [She02].

In order to overcome such a difficulty, a different family of approaches [Wu05], [She03], [Asi05], [Li07], [Qiu09] consists of directly parameterizing closed genus-0 meshes onto a spherical domain (*i.e.*, unit sphere), since such objects are topologically equivalent to a sphere.

Thus, the spherical parameterization problem is considered as an embedding of the model in the unit sphere. All the mesh vertices will lie on the sphere's surface and the condition to be satisfied in order to obtain a valid spherical parameterization is to ensure that the resulting spherical triangles are non-overlapping.

In practice, determining valid spherical parameterization proves to be a more challenging task than the planar case.

Historically, one of the very first spherical parameterization techniques proposed was introduced in [Ken92]. The method returns a valid embedding only if the original mesh has a convex shape. A convex model has the property that any two vertices can be connected by a straight line segment which lies inside the model and does not intersect the shape. In this case, the parameterization becomes quite simple. The model is first translated so that its centroid coincides with the origin of the given coordinate system and then the vector position of each vertex is normalized to unity. As a result, all the vertices will lie on the unit sphere surface.

This simple spherical projection can be extended to the class of so-called star-shaped models. Such objects have the property that there is at least one point in the interior of the model which can be connected with all the mesh vertices by a straight line without generating multiple intersections with the mesh surface. The only problem is to determine the interior points (called also kernel points) which satisfy such properties. In the case of star-shaped objects, the kernel can be determined as the intersection of all semi-spaces defined by the set of mesh faces.

Such simple approaches are illustrated in Figure IV 19. For simplicity, we have illustrated here the 2D case (*i.e.*, closed and planar polygons parameterized onto the unit disk). Figure IV 19.a presents the case of a convex polygon. The blue point represents the corresponding gravity center. Figure IV 19.b illustrates the case of a star shape polygon. The center of the unit disk is here placed in an arbitrary position, which does not correspond to a kernel point. As a result, the resulting parameterization is not valid (overlapping arcs on the unit disk represented in red). In contrast, in Figure IV 19.c, for the same star shape polygon, the center of the unit disk is placed into a kernel point. As a result, the obtained parameterization is in this case valid.

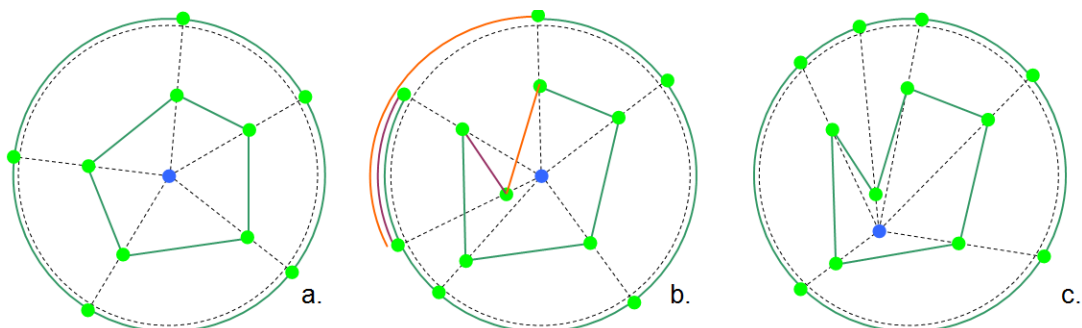


Figure IV 19. Shape projection on a circle. (a) Kent method applied on a convex shape; (b) Kent method applied on a non convex shape; (c) kernel approach.

Let us note that such simple approaches do not optimize neither angle nor area distortions, since no geometric information is taken into account. In addition, they can be useful for

simple shapes but in the case of real life objects, the assumptions of convexity or star shape do not hold.

Starting from the Kent *et al.* [Ken92] approach, Alexa [Ale00] develop a method to project any kind of 3D genus-0 meshes onto the unit sphere. In order to deal with the triangle overlapping problem, the author introduces a vertex relaxation process which consists of an iterative procedure that repeatedly places each vertex at the center of its neighbors. Since the new vertex position is not on the sphere, a normalization operation is required:

$$\varphi_i^{l+1} = \frac{\sum_{j \in \text{Neighbors}(i)} \varphi_j^l}{\left\| \sum_{j \in \text{Neighbors}(i)} \varphi_j^l \right\|}, \quad (\text{IV.50})$$

where φ_i^{l+1} is the position, in the parameter domain, of the vertex φ_i^l after the $(l+1)$ relaxation iterations. The process continues until the largest displacement of any mesh vertex becomes smaller than a predefined threshold.

However, the relaxation process can lead to mesh collapses into a single point. In order to solve this problem, it is necessary to fix several vertices (called anchors) in the parametric domain. Unfortunately, without a sufficient number of adequately selected anchors the embedding may also collapse, as illustrated in Figure IV.20.a. Here, 4 anchor points have been considered.

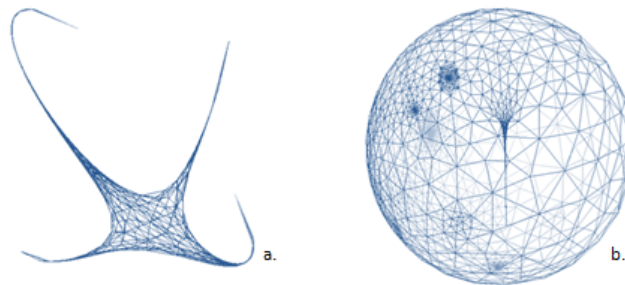


Figure IV.20. Problems encountered in sphere parameterization: (a) Collapsed mapping; (b) Overlapped triangles in sphere parameterization [Ale00].

In addition, because the position of the anchors is fixed, this can lead to triangle overlapping (Figure IV.20.b). In order to solve the problem, a heuristic scheme is developed which consists of changing the anchor points after a given number of iterations. The relaxation process is illustrated in Figure IV.21 for a 3D model representing a horse.

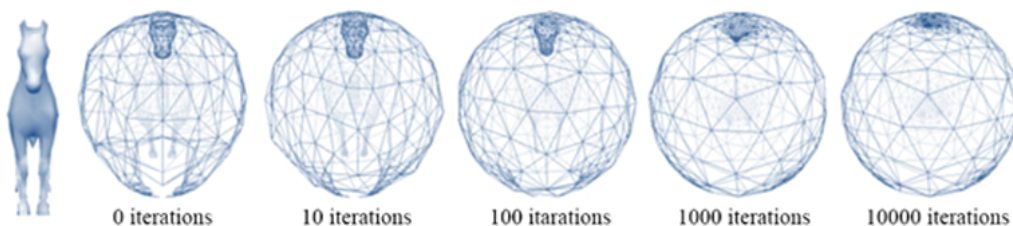


Figure IV.21. Spherical parameterization using relaxation approach proposed by Alexa[Ale00].

The method provides valid parameterization in a majority of cases. However, the relaxation process is difficult to be controlled and does not guarantee a valid embedding in all cases.

Another interesting method starts by reconsidering the principle of virtually cutting the mesh through a path in order to obtain an artificial boundary [Hak00] (Figure IV.22). A conformal planar parameterization is first computed using an arbitrary mesh triangle as a boundary. Then, a stereographic projection of the resulting planar mapping is performed in order to obtain the spherical parameterization. The boundary triangle will represent the north pole of the sphere.

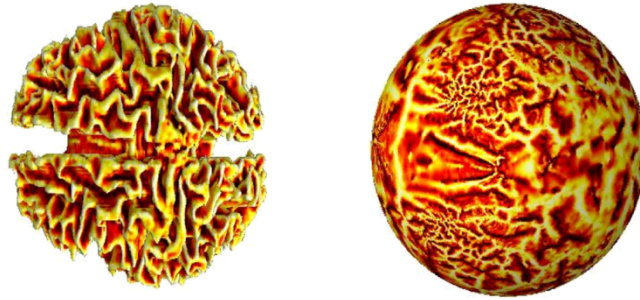


Figure IV.22. Spherical parameterization using the Haker's approach [Hak00].

The approach offers the advantage of simplicity since authors construct only a sparse, real, symmetric, linear system of equations.

Despite its simplicity, the conformal surface parameterization proposed by Haker [Hak00], presents some important drawbacks. First of all, the results are strongly influenced by the choice of the boundary triangle. In addition, all the vertices of the mesh tend to cluster in the center of considered triangle leading to a significant area distortion in the parameter domain. Finally, the inverse stereo projection technique does not preserve the geometric properties of the planar triangulation so the result will be even more distorted. Finally, and more important, the method generally suffers from foldovers.

Extending the idea of opening the genus-0 meshes in order to map them with the help of existing planar parameterization methods, various approaches cut the mesh into two parts, each topologically equivalent to a disk. The two parts are parameterized each over a planar disk with a common boundary, and then each disk is mapped onto a hemisphere of the unit sphere.

One of the first approaches based on this principle is proposed in [Ise01]. A so-called *vertex separator* algorithm is here proposed which partitions the mesh into two components with approximately equivalent numbers of vertices and with a common boundary. A modified version of the Tutte parameterization [Tut63] is used for parameterizing each of the two components. Then the two planar embeddings are mapped onto the sphere using the stereographic projection. Naturally, the results strongly depend on the considered cut.

More recently, starting from the same idea, Jianping Hu *et al.* [Hu08] present a similar method. The difference here concerns a novel splitting technique which cuts the mesh into two parts based on the optimum reflective plane approach introduced in [Kaz03]. After the mesh partition is achieved, an initial planar parameterization as the one proposed by Floater in [Flo03] is applied for the two mesh pieces. In addition, a spherical stretch optimization is performed in the parametric domain.

Within the same family of approaches, Zayer [Zay06] propose a method which re-formulates the problem in a curvilinear coordinates system (*i.e.* spherical coordinates with radius = 1), hence reducing it to a 2D problem where the each vertex position is represented by the azimuth angle (longitude) $\theta \in (0, 2\pi)$ and the elevation angle (latitude) $\phi \in (0, \pi)$. In order to eliminate the pole singularity problem, the two poles of the model are determined and removed.

The shortest path between the two poles, called date line is then determined with the help of the Dijkstra algorithm [Dij59]. This path is then used for cutting the mesh, which yields an open mesh. In this manner, an initial harmonic map as proposed in [Eck95] or the mean coordinates value approach [Flo03] can be next applied to obtain the corresponding planar parameterization.

In order to reduce the distortions particularly at triangles located near the cut path, an optimization step is performed based on the tangential Laplacian operator. The various steps of the methods are illustrated in Figure IV.23.

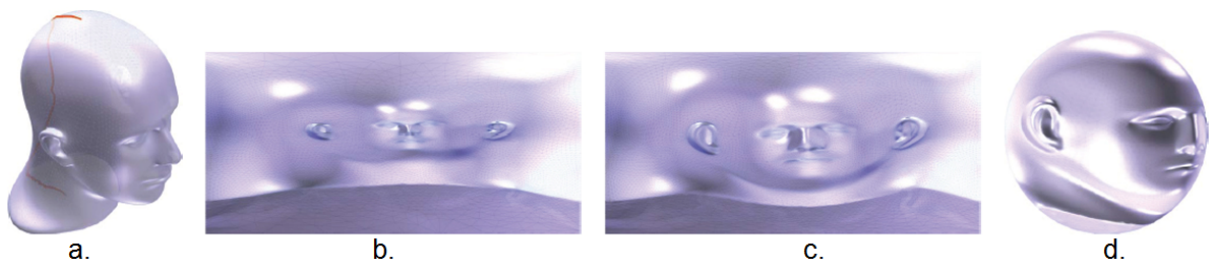


Figure IV.23. Curvilinear Spherical Parameterization [Zay06]. (a) mesh cut along the date line; (b) the initial parameterization in curvilinear coordinates (with high distortions); (c) the improved mapping taking into account spherical distortion; (d) the final spherical parameterization.

Gotsman *et al.* [Got03] propose an extension of the theory of barycentric coordinates used in the planar mapping case to the spherical case. However, such an extension requires a transition from a linear to a non-linear framework. Thus, in order to embed a closed mesh with n vertices into the unit sphere, a positive weight w_{ij} is defined for each edge $\{i, j\}$. The procedure leads to the following set of $4n$ non-linear equations with $4n$ unknowns for the embedding coordinates $\varphi_i(x_i, y_i, z_i)$ and auxiliary variables α_i :

$$\begin{cases} x_i^2 + y_i^2 + z_i^2 = 1 \\ \alpha_i x_i - \sum_{j \in \text{Neighbors}(i)} w_{ij} x_j = 0 \\ \alpha_i y_i - \sum_{j \in \text{Neighbors}(i)} w_{ij} y_j = 0 \\ \alpha_i z_i - \sum_{j \in \text{Neighbors}(i)} w_{ij} z_j = 0 \end{cases} \quad (\text{IV.51})$$

The auxiliary variables α_i are real numbers that are introduced in order to simultaneously solve the system in its null space.

Similarly to the planar case, the characteristics of the parameterization can be controlled by the weights w_{ij} . Starting from the above system of equations, Gotsman *et al.* [Got03] obtain different parameterizations using various weights: the uniform Tutte weights, the cotangential weights for a conformal angle-preserving mapping proposed by Eck *et al.* [Eck95] or those introduced by Desbrun [Des02].

In [Got03] the authors do not provide an efficient way to solve the resulting system (IV.51). Using generic non-linear solvers can lead to a prohibitive computational cost, which limits the applicability of the method to meshes described by a high number of faces/vertices.

More recently, Saba *et al.* [Sab05] have proposed a solution to efficiently solve such a system. Their approach breaks down the problem into a two-step procedure involving two systems of equations, one linear and one non-linear. The linear system is solved using a multiresolution algebraic multigrid approach and its solution is used as an initial guess for solving the nonlinear system.

In order to generate the initial guess, Saba *et al.* [Sab05] use a variant of the method proposed by Isenburg *et al.* [Ise01]. They first partition the mesh into two balanced sub-meshes and then embed each sub-mesh in a planar disk using the barycentric method with weights w_{ij} . Next, the two planar parameterizations with the common boundary are mapped onto the sphere using the inverse stereographic projection. The nonlinear system presented in equation (IV.51) is finally solved using a variation of the Gauss-Seidel method.

An interesting method which directly parameterizes a closed genus-0 mesh on the unit sphere is the one proposed by Sheffer *et al.* in [She03]. They present an algorithm for spherical embedding that extends the planar mapping approach in [She01]. Authors formulate the following set of necessary and sufficient conditions for the angles to form a valid spherical triangulation:

$$\begin{cases} \alpha_i^j > 0 \\ e_i > 0 \\ e_i < 2\alpha_i^j \\ \alpha_i^0 + \alpha_i^1 + \alpha_i^2 - e_i - \pi = 0 \\ \sum_j \sum_i \alpha_i^j - 2\pi = 0 \end{cases} \quad \begin{matrix} i = 1, \dots, N_F \\ j = 0, 1, 2 \\ k = 1, \dots, N_v \end{matrix} \quad (\text{IV.52})$$

where: α_i^j represent the j spherical angle and e_i the spherical excess of the i^{th} triangle. N_v and N_F denote the number of mesh vertices and faces respectively. The excess of a triangle is defined as the area of the region on the sphere determined by that triangle.

This set of equations and conditions can be transformed into a constrained minimization problem, where the least-squares distance of the solution values (α_i^j and e_i) from their target values (β_i^j and e_i') are minimized:

$$F(\alpha, e) = \sum_i \sum_j (\alpha_i^j - \beta_i^j)^2 + \sum_i (e_i - e_i')^2 \quad (\text{IV.53})$$

The energy function in equation (IV.53) allows to control the shape of the parameterization by optimizing spherical angles and/or area values. For example, if a conformal mapping is required, the target values of angles can be set to be equal to the angles in the initial object space. On the contrary, if an equiareal mapping is required, then the target values for areas are set equal to the areas of triangles in the original space.

Once the parametric angles α_i^j are determined, the spherical triangulation may be generated. Starting from an arbitrary triangle, a vertex is fixed on the sphere and then according with formulas from spherical trigonometry the remaining triangle vertices are placed accordingly. Thus, the lengths of triangle edges (a , b and c) are computed using the cosine rule:

$$a = \arccos\left(\frac{\cos A + \cos B \cos C}{\sin B \sin C}\right) \quad (\text{IV.54})$$

where A , B and C are angles in the considered triangle opposed to a , b and c . Lengths b and c are computed similarly with a . Based on the first fixed vertex arbitrary position, the previous computed edge lengths and the triangle angles, the 2D position of the other two vertices is straightforward. Then the vertex positions for the neighboring triangles of the first chosen triangles are determined.

Unfortunately, it seems that the spherical formulation is numerically much less stable than its planar equivalent [She01]. For this reason, it can be applied only for meshes with less than a few hundreds of vertices.

A multiresolution technique, inspired from the approaches introduced in [Sha98] and [Hor99], is proposed in [Pra03]. The original mesh is first simplified by applying a sequence of vertex removal operations, until a tetrahedron is obtained. The tetrahedron is then simply projected onto the unit sphere. Next, the vertices are inserted into the sphere in a progressive mesh sequence constructed with the help of a vertex split operation.

Each vertex split specifies a ring of vertices that represents the neighbors of the new vertex to be inserted. In order to obtain a valid embedding (with no overlapped triangles), the new vertex will be placed inside of the spherical polygon described by his neighbors. An optimization procedure is applied in order to minimize the stretch metric of the parameterization.

A similar approach is proposed in [Bir04], where the mesh is also simplified to a tetrahedron. The tetrahedron is then mapped onto a spherical surface and afterwards the simplification process is reversed by iteratively inserting the vertices on the surface of the sphere. The process optimizes the position of each new inserted vertex until it becomes the barycenter of its neighbors. This procedure yields an initial parameterization. This initialization is then optimized in order to preserve as well as possible the angles between the edges and the ratio of the edge-lengths. This is achieved by minimizing the weighted square sum of angles to all neighboring vertices:

$$\sum_{j \in \text{Neighbors}(i)} \arccos(\varphi_i \cdot \varphi_j)^2 w_{i,j} \quad (\text{IV.55})$$

where φ_i represents the considered vertex position in the parametric domain and φ_j belongs to its adjacent neighbors. The edge weights $w_{i,j}$ can be chosen as uniform [Tut63] or as the mean value coordinates of Floater [Flo03].

Despite the various optimizations involved, the resulting mapping suffers from high distortions. This limitation is illustrated in Figure IV.24, for two different textured meshes.

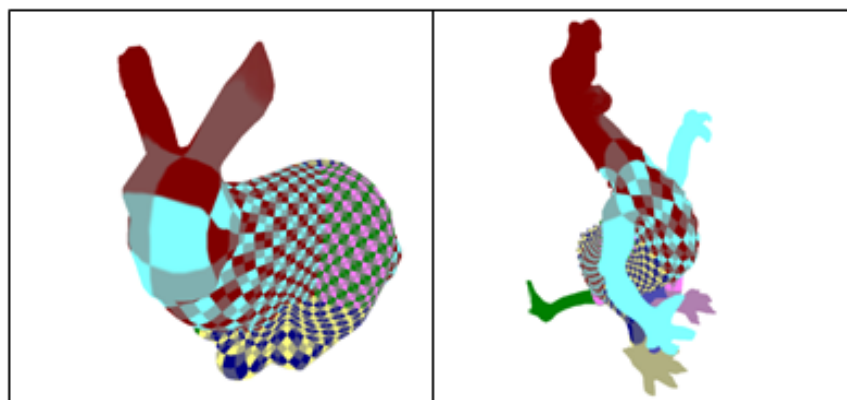


Figure IV.24. Two textured meshes after a spherical parameterization with Birkholz approach [Bir04].

In addition to planar and spherical parameterization methods which are only applicable to surfaces with disk topology and genus zero meshes respectively; there are also approaches that treat models with arbitrary genus. Generally, the process of parameterization is done by first segmenting the mesh into disk-like patches which are then mapped into the planar domain as illustrated in Figure IV.25. This consists of defining a so-called atlas of parameterizations.

Within the framework of parameterization methods, the involved segmentation techniques aim at partitioning the surface into a set of patches such that the parameterization distortions of each patch are minimized. An additional constraint requires to keep a low number of patches with associated boundaries as short as possible. Since planar patches are by definition developable, one possible approach is to segment the surface into nearly planar patches [San01], [Mai93], [San03].

The main challenge in this case concerns determining mappings that are smooth across the patch boundaries. The first methods proposed in this area [Pra01], [Gus02], [Flo02b], are penalized by this problem. However, there are some solutions [Gu03], [Kho03]) which guarantee a globally smooth parameterization with only a few singularities points.

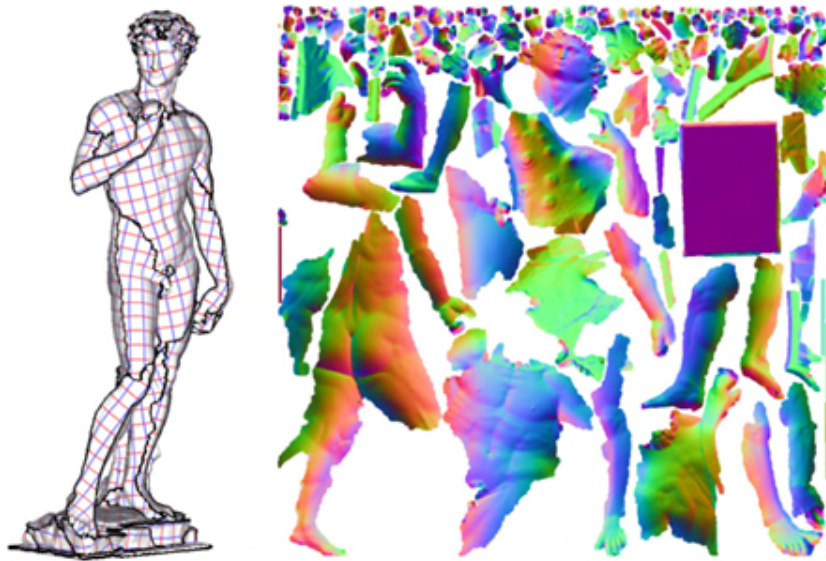


Figure IV.25. Planar parameterization of meshes with arbitrary genus.

Such a type of parameterization is particularly adapted for various applications, including texture mapping, compression or remeshing. However, because of the relatively high number of patches which can be generated and the uncontrolled shape of the boundaries, it is not suitable for morphing applications, where a correspondence between patches and vertices of two models needs to be determined.

The various spherical parameterization techniques described in this section, with corresponding properties are summarized in Table IV.3 and Table IV.4 .

Table IV.3. Comparison of spherical parameterization methods – Part 1.

Method	Type of distortion minimized	Complexity	Computational time	Comments
[Ken92] Kent	None	+	+	<ul style="list-style-type: none"> - applicable for genus-0 meshes - bijective mapping only for convex models - do not preserve the shape of the mesh - the parameterization is made directly on the unit sphere
[Ale00] Alexa	None	++	+++	<ul style="list-style-type: none"> - applicable for any genus-0 meshes - an iterative process is performed to position each vertex at the center of its neighbors - require to fix some vertices on the sphere (which are changed after few relaxation steps) - the method not guarantees a valid embedding all the time
[Hak00] Haker	Angles	++ (Linear)	+	<ul style="list-style-type: none"> - applicable for any genus-0 meshes - eliminate a triangle from the mesh to create a virtual boundary - make the parameterization into the plane of the resulting open mesh and then using stereo projection technique embed the model on the sphere - the stereographic projection technique does not preserve the shape -the mapping depends heavily on the eliminated triangle - the result is not bijective
[Hu08] Hu	Angles	++ (Linear)	+	<ul style="list-style-type: none"> - split the model into two pieces based on the optimum reflective plane - make a planar parameterization based on the Floater [Flo03] method - implement an optimization technique for the vertices along the cut (since here are the biggest distortions)
[Zay06] Zayer	Angles	++ (Linear)	+	<ul style="list-style-type: none"> - the parameterization is performed on the curvilinear coordinates - set two vertices (poles) then cut the model between this points and open it - apply a planar parameterization (a modification of the Floater [Flo03] approach) - implement an optimization step in the curvilinear coordinates along the cut
[Got03] Gotsman	Angles or Area or Distances	+++++ (Non Linear)	+++++	<ul style="list-style-type: none"> - extend the barycentric coordinates used in planar parameterization, but this lead to a non-linear problem - can implement different weights depending on the type of distortions to be minimized - for meshes with a high number of vertices the problem remain unsolved
[Sab05] Saba	Angles or Area or Lengths	+++ (Non Linear)	++++	<ul style="list-style-type: none"> - find a solution for the [Got03] method using a multigrid computational approach - for an initial embedding, they partition the mesh into two pieces and parameterize each one into the plane, then implement the stereographic projection - the parameterization do not guarantee to be always bijective
[She03] Sheffer	Angles and/or Areas	++++ (Non Linear)	+++++	<ul style="list-style-type: none"> - extend the idea present in [She01] for spherical case - the parameterization is made directly on the unit sphere - the triangles angles are first determined, then the vertices are placed on the sphere one by one - the method is not stable

Table IV.4. Comparison of spherical parameterization methods – Part 2.

Method	Type of distortion minimized	Complexity	Computational time	Comments
[Pra03] Praun	Lengths	++++ (Non Linear)	+++++	<ul style="list-style-type: none"> - the method is based on a multiresolution technique which reduce the model to a tetrahedron - the tetrahedron is then simple map on the sphere, and then the vertices are reintroduced in a progressive mesh sequence - the result is a bijective mappings
[Bir04] Birkholz	Areas and Lengths	++++ (Non Linear)	+++++	<ul style="list-style-type: none"> - the method is similar with [Pra03], but additionally it introduces an algorithm which tries to preserve the angles and the ratio of the edge-lengths - the optimization algorithm proves not to be so good, resulting a very distorted parameterization
[Kha06] Kharevych	Angles	+++ (Non Linear)	++++	<ul style="list-style-type: none"> - free boundaries or user controlled boundary shape via prescribed curvatures - map meshes with arbitrary topology to the plane - meshes of genus zero can be parameterized over the sphere - the parameterization can contain global overlaps

The next section introduces the curvature-driven spherical parameterization method proposed.

IV.3.2. Curvature-driven spherical parameterization

The proposed spherical parameterization method is dedicated to closed 3D, genus-0, two-manifold meshes. The main principle consists of exploiting the Gaussian curvature in order to jointly minimize length, angular and area distortions.

Let us first briefly recall definitions of a 3D surface Gaussian curvature and describe how such a measure can be computed in the case of 3D meshes.

IV.3.2.1. Theoretical aspects

In the 2D Euclidian space the curvature of a planar curve can be interpreted as a measure of the local variation to the curve's tangent in a given point. Such a measure provides the amount by which the considered curve deviates from a straight line (Figure IV 26.a).

If we consider a planar curve C , at a given point $p \in C$, the curvature value can be determined as the inverse radius r of an osculating circle O_C . A larger value for the osculation circle radius implies a smaller magnitude for the curvature. A straight line has zero curvature.

In the case of surface in the \mathbb{R}^3 space, the notion of curvature becomes more complex. Let us consider a point p on a continuous and C^2 smooth surface S defined in the \mathbb{R}^3 domain. If

we take the surface intersection with the family of planes pathing through p and including the normal vector at point p with the surface S , we obtain a family of 2D curves (Figure IV 26.b).

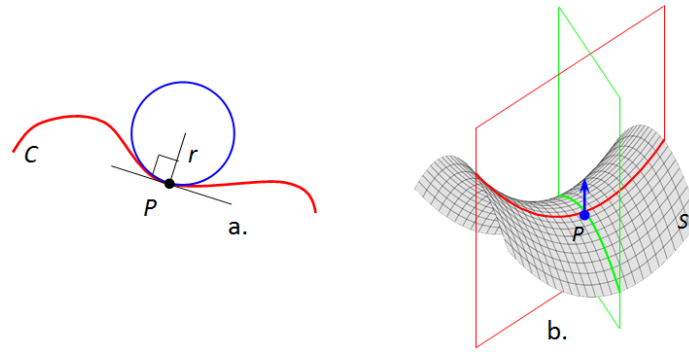


Figure IV 26. The curvature in a point for a: (a) curve; (b) 3D surface.

For each of them a curvature value can be determined. The minimum and maximum values are called *principal curvatures* and are denoted by k_1 and k_2 .

Based on these norms, two types of measures can be computed: Gaussian (K) and mean (H) curvature, defined by the following equations:

$$K = k_1 \cdot k_2 \quad \text{and} \quad H = \frac{k_1 + k_2}{2} \quad (\text{IV.56})$$

Curvature measures are by definition expressed as functions of the second order surface derivatives. Thus, they are associated with smooth, C^2 -continuous surfaces. However, 3D meshes are at most C^0 -continuous surfaces and do not fulfill the smoothness conditions required. In this case, it is necessary to perform a piecewise linear approximation in order to obtain an approximation of the of the curvature values.

In our work, we have adopted the approximation technique introduced in [Zxu09], recalled here-below.

For a vertex p of a mesh M , let $\{p_i \in \text{Neighbors}(p) \mid i=1, 2, \dots, l\}$ be the set of the one-ring neighbor vertices and $\{(p_i p p_{i+1}) \in F \mid i=1, 2, \dots, l\}$ the set of adjacent triangles. If we denote by α_i the angle determined by p_i , p , and p_{i+1} , then we can compute the angular defect at the point p as:

$$\delta = 2\pi - \sum_i \alpha_i \quad (\text{IV.57})$$

The Gaussian curvature is then defined as described by the following equation:

$$K = \frac{2\pi - \sum_i \alpha_i}{G} \quad (\text{IV.58})$$

where G is a geometrical factor directly correlated with the model. We have adopted the approach proposed in [Zxu09] where G is selected as:

$$G = \sum_i A_i(p) / 3 \quad , \quad (\text{IV.59})$$

where $A_i(p)$ denotes the areas of triangles adjacent to vertex p . We obtain thus, the following approximation for the Gaussian curvature:

$$K = \frac{3(2\pi - \sum_i \alpha_i)}{\sum_i A_i(p)} \quad (\text{IV.60})$$

The next section describes how this curvature measure is exploited for spherical parameterization purposes.

IV.3.2.2. Core algorithm

The proposed parameterization algorithm consists of the following three core steps:

- **Step1** - *Curvature-driven iterative flattening*

First, we compute the Gaussian curvature K_p for each vertex p of the mesh. Then, we determine the vertex p_{max} with the maximum absolute value of the Gaussian curvature. The barycenter of its neighboring nodes is computed, as described by the following equation:

$$p'_{max} = \frac{\sum_{p_i \in Neigh(p_{max})} p_i}{val(p_{max})}, \quad (\text{IV.61})$$

where $Neigh(p_{max})$ denotes the set of vertices adjacent to p_{max} and $val(p_{max})$ represents its valence.

If the Euclidian distance between new and initial positions $\|p'_{max} - p_{max}\|$ is superior to a threshold $dist$, its position is changed to p'_{max} . Otherwise, the considered vertex is not affected and the algorithm selects as a candidate the following highest curvature vertex, reiterating the process.

When modifying the position of a vertex, the various measures (triangle areas and angles) involved in the computation of the Gaussian curvatures, need to be re-computed. This is done locally, exclusively for the displaced vertex and for its neighbors, since the other mesh vertices are not affected.

This process is recursively repeated:

1. Determine the vertex with maximum Gaussian curvature,
2. Compute the barycentric coordinates,
3. Displace the vertex and re-compute Gaussian curvature K only for the affected vertices.

In this manner, salient mesh vertices are firstly detected and processed, leading, after each iteration, to a locally flattened version of the 3D mesh model. At the end of the process, a sphere-like surface is obtained. In contrast with [Ben08] that also use the Gaussian curvature

in parameterization purposes, in order to identify the high-curvature vertices and concentrate the entire mesh curvature there, our goal is to determine such vertices in order to distribute the curvature to its neighbors and thus construct models with constant curvature values, like the unit sphere.

This process is illustrated in Figure IV.27, which presents the evolution of a given 3D mesh after a certain number of iterations.

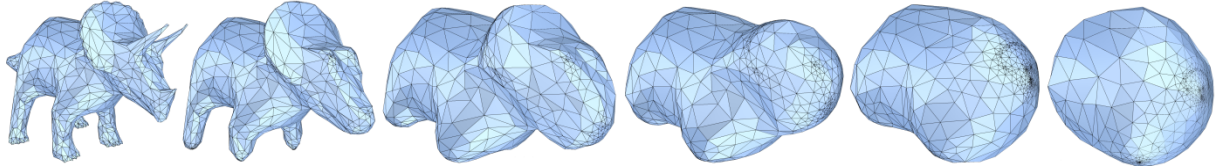


Figure IV.27. Iterative curvature-driven flattening.

The Gaussian curvature in equation (V.60) privileges the selection of vertices located in densely sampled mesh regions, where the triangle areas tend to zero. Unfortunately, this behavior can penalize our algorithm, which can perform long sequences of iterations inside such regions.

In order to avoid such a problem, we have considered a modified expression of the Gaussian curvature, defined as:

$$K_s = \frac{2\pi - \sum_i \alpha_i}{\chi_s + \sum_i A_i(p)/3} \quad , \quad (\text{IV.62})$$

where χ_s denotes the average triangle area, computed over the entire mesh. The correction factor χ_s makes it possible to reinforce, in the selection process, the influence of the angular defect term $(2\pi - \sum_i \alpha_i)$ and thus to avoid long loops in densely sampled regions characterized by low values of triangle areas.

Step 1 is successively repeated for a number It of iterations. In practice, the usual value for parameter It is set to be five times the number of vertices.

At the end of step 1, a size normalization process is applied in order to avoid shrinkage problems (*i.e.*, the model is centered at the origin and the maximum distance between any vertex and the origin is used to normalize the vertices positions).

- **Step 2** - *Visibility check and projection onto the sphere*

At this stage, we first check if the mesh obtained at the end of step 1 can be stereographically projected onto the sphere. This consists in applying for each mesh vertex a visibility test performed with the help of a ray casting operation. If all mesh vertices are visible

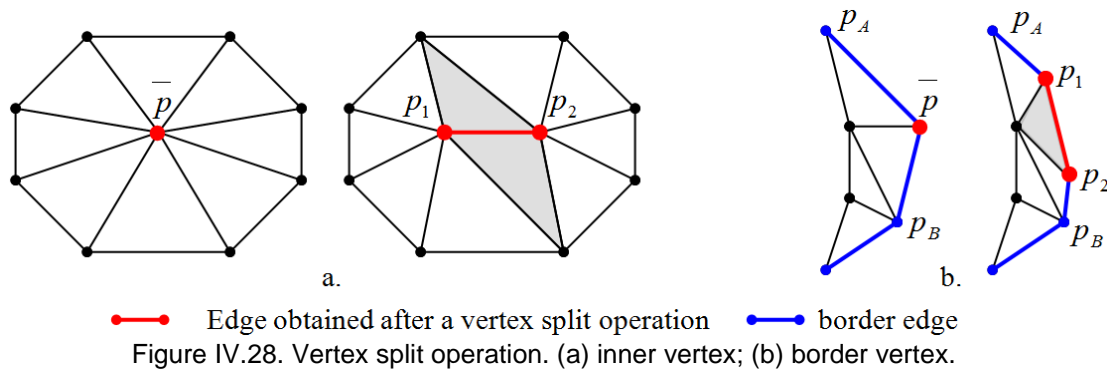
from the object's gravity center, the mapping onto the unit sphere is simply obtained by a vertex projection defined as described by the following equation:

$$\forall p_i \in V, \quad \phi_i = \frac{P_i}{\|P_i\|} \quad (\text{IV.63})$$

where ϕ_i is the image on the unit surface sphere of the vertex p_i . The visibility property ensures that the obtained parameterization is bijective. If the visibility condition is not satisfied, then step 1 is re-iterated.

- **Step 3 - Vertex split sequence**

Here, all the vertices removed in the mesh simplification are iteratively re-inserted on the sphere by constructing a progressive mesh sequence analogously to the method described in [Hop93] by Hoppe and Praun. The algorithm exploits the fact that a contraction operation is invertible. For each edge collapse, a corresponding inverse operator, called vertex split, is defined (Figure IV.28).



Thus, starting from a coarser version of a 3D model together with a series of records, indicating how it was simplified, we can produce a sequence of intermediate models applying a series of vertex split operations until we reach the original object. Normally, this requires for each item in the split sequence to encode the vertex being split \bar{p} , positions for the two initial vertices p_1, p_2 and all the original adjacencies.

In contrast with Hoppe and Praun [Hop93] objectives that try to reconstruct the original shape of the model from a coarser version of it, we aim to return to the original mesh topology with its surface directly mapped on the sphere. Thus, in our case, in the mesh simplification process, we will store only the vertex obtained after each edge collapse operation, the two original edge endpoints and the corresponding adjacencies. When employ a vertex split operation, the positions of p_1 and p_2 must be computed accordingly with the adjacent vertex coordinates.

Additionally, the objective is to re-insert a removed vertex in the mesh structure without generating triangle flipping or degenerate faces. This requires a position optimization of the

vertex to be inserted. In contrast to the approach in [Pra03], which implements also a parameterization technique based on mesh simplification followed by a vertex split process, we have adopted a simple, yet efficient optimization procedure, illustrated in Figure IV.29.

The first ring neighborhood from which the considered vertex was removed (Figure IV.29.a) is first subdivided in order to obtain a set of potential positions (Figure IV.29.b) where the vertex to be inserted. Each face is split after a 1-to-4 triangles scheme and 3 levels of such subdivisions are performed.

A sub-set of valid possibilities (*i.e.*, position which do not lead to overlaps or degenerate triangles) is then determined (Figure IV.29.c). In order to accomplish this task, we establish for each potential position if the new edges that would form intersect the boundary edges defined by the first ring neighborhood. If no intersection is produced then the position is considered valid.

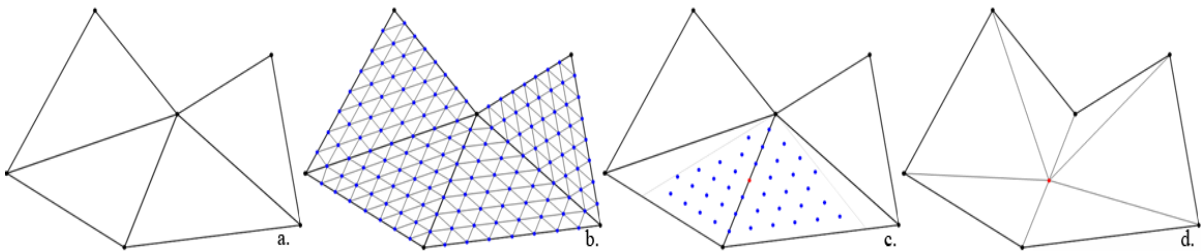


Figure IV.29. Vertex insertion operation: (a) initial configuration; (b) polygon subdivision; (c) set of valid positions; (d) final retained position and the new configuration.

Among them, the vertex which provides the optimal angular distribution of the corresponding triangles is determined (Figure IV.29.d). In order to reach this objective, we select the position which yields the maximal value of the minimal angle of the adjacent triangles.

Let us note that if the mapping is an embedding prior the vertex split operation, then it should remain also valid after the insertion.

IV.3.3. Experimental evaluation

This section aims to provide several experimental results regarding the performance of our spherical mesh parameterization algorithm based on surface Gaussian curvature. In order to validate the proposed algorithm we have considered from the Princeton Shape Benchmark and MPEG 7 database a set of eight closed, manifold, triangular mesh models characterized by various types of geometries, complexities and shapes.

We pre-process each 3D model using with the modified version of the QME mesh simplification technique introduced in [Gar97], in order to reduce the total number of mesh vertices and thus to considerably decrease the computational complexity.

Figure IV.30 and Figure IV.31 present some results obtained after applying the proposed algorithm, with the various intermediate stages involved. In all cases, the obtained spherical parameterizations yield valid embeddings which preserve well the shape of the test models.

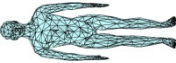

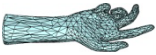
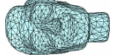


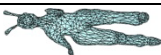

In order to objectively evaluate our approach, we have accomplished a comparative analysis of our implementation and the ones proposed by Alexa [Ale02] and Praun *et al.* [Pra03] in terms of angle (AD) and area (SD) distortions. AD and SD are defined as described by the following equations [Yos04]:

$$AD = \frac{1}{3F} \sum_{i=1}^F \sum_{j=1}^3 |\alpha_{ij} - \alpha'_{ij}| \quad (IV.64)$$

$$SD = \sum_{i=1}^F \left| \frac{A(T_i)}{\sum_{j=1}^F A(T_j)} - \frac{A(T'_i)}{\sum_{j=1}^F A(T'_j)} \right| \quad (IV.65)$$

Ideally, both distortions should be as close as possible to zero, which correspond to the case when all mesh triangles remain unmodified after the mapping process. The results synthesized in Table IV.5 show that the proposed method provides superior performances in terms of both angular and area distortions, with gains of 36,72% and 19,04% respectively when compared to Alexa's method and gains of 35,85% for angular distortions and 19,55% for area distortions compared with Praun's method.

Table IV.5. Comparative study concerning area and angle distortions.

Name	Model	No. of vertices	Proposed method		Alexa method [Ale02]		Praun <i>et al.</i> method [Pra03]	
			AD	SD	AD	SD	AD	SD
Man		14603	0.454	1.417	0,793	1,431	0,651	1,452
Lyon		956	0.371	1.174	0,512	1,388	0,445	1,413
Hand		25001	0.353	1.126	Overlapping		0,573	1,538
Face		17358	0.347	0.576	0,456	0,933	0,521	0,775
Horse		19851	0.391	1.194	0,803	1,636	0,637	1,726
Rabbit		453	0.311	0.682	0,362	0,891	0,364	0,782
Alien		16266	0.368	1.288	Overlapping		0,872	1,673
Dino		16995	0.384	1.417	0,962	1,552	0,896	1,728

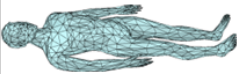
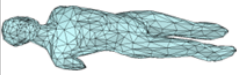
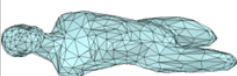
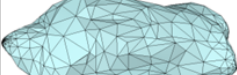


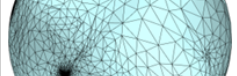







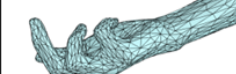
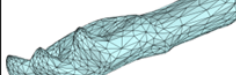
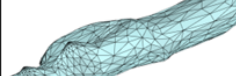
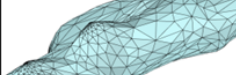

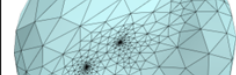
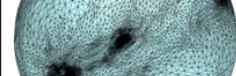






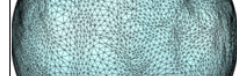
Original Models	Intermediary steps					Spherical Parameterization (simplified version)	Spherical Parameterization
							
Man Model – Original: 14603 Vertices and 29202 Triangles; Simplified: 1000 Vertices and 1996 Triangles							
							
Lion Model – Original: 956 Vertices and 1908 Triangles; Simplified: 500 Vertices and 996 Triangles							
							
Hand Model – Original: 25001 Vertices and 49998 Triangles; Simplified: 1000 Vertices and 1996 Triangles							
							
Head Model – Original: 17358 Vertices and 34712 Triangles; Simplified: 900 Vertices and 1796 Triangles							

Figure IV.30. Visual evaluation of our 3D mesh spherical parameterization (1).

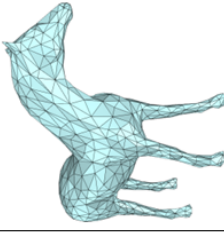
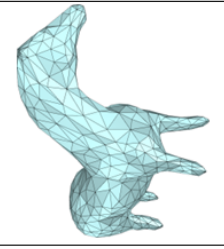
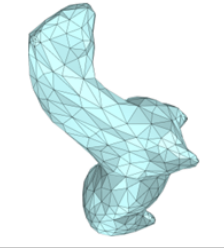
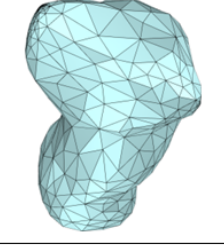
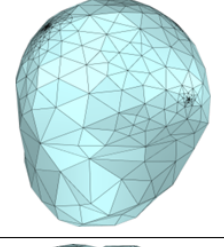
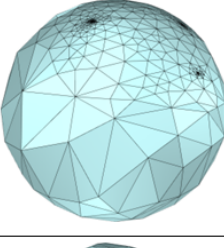
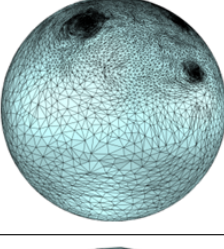
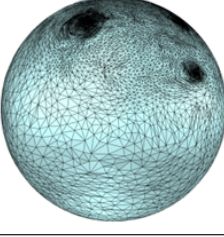
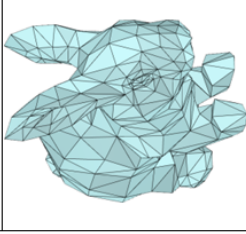
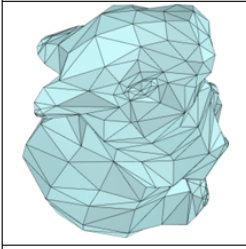
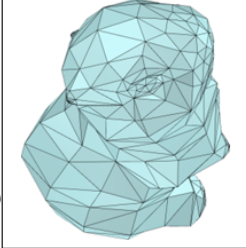
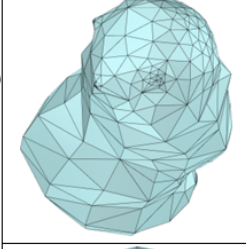
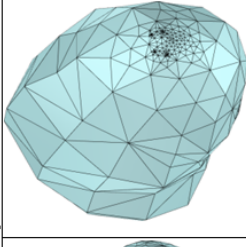
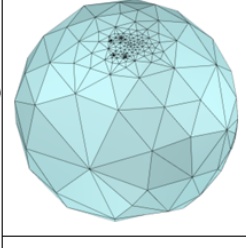
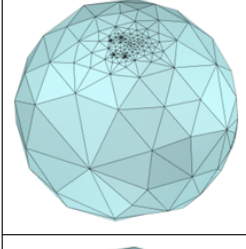
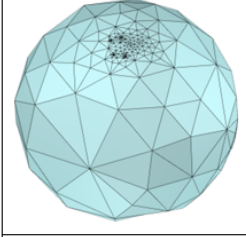
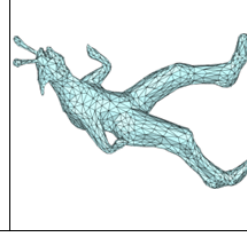
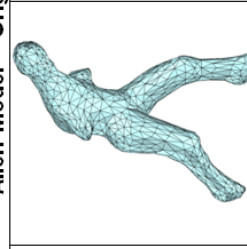
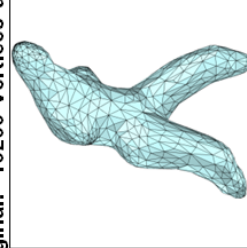
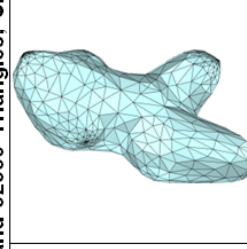
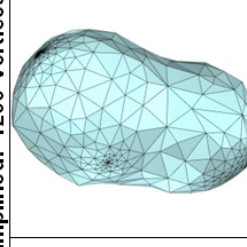
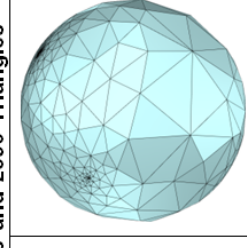
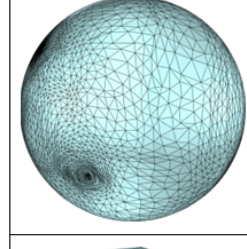
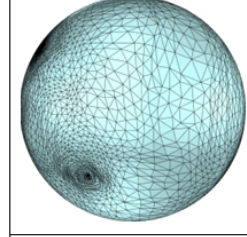
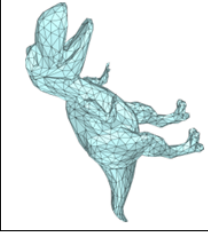
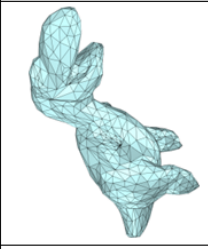
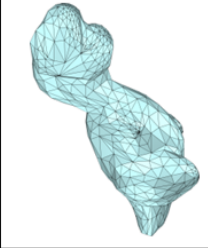
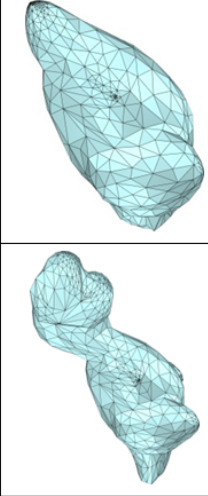
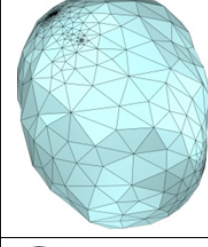
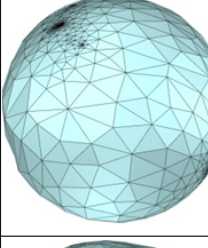
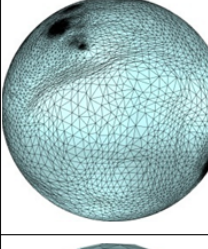
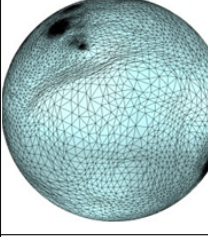
Original Models	Intermediary steps					Spherical Parameterization (simplified version)	Spherical Parameterization
							
Horse Model – Original: 19851 Vertices and 39698 Triangles; Simplified: 800 Vertices and 1596 Triangles							
							
Rabbit Model – Original: 453 Vertices and 902 Triangles; Simplified: 453 Vertices and 902 Triangles							
							
Alien Model – Original: 16266 Vertices and 32530 Triangles; Simplified: 1200 Vertices and 2396 Triangles							
							
Dino Model – Original: 16995 Vertices and 33988 Triangles; Simplified: 1100 Vertices and 2196 Triangles							

Figure IV.31. Visual evaluation of our 3D mesh spherical parameterization (2).

Regarding the processing requirements, the proposed algorithm is slightly slower than the other two approaches. This is due to the iterative computation of the Gaussian Curvature. In contrast, the approach proposed by Alexa projects directly the vertices onto the sphere employing simple vertex normalizations operations and different relaxation processes. However, Alexa's technique does not guarantee a valid embedding for all the models.

Concerning the Praun's method, the mesh simplification process is here performed until a simple tetrahedron, which can be directly projected onto the unit sphere. Despite the optimization procedure employed when re-inserting the initial mesh vertices, the resulting distortions are here more important. This shows the interest of stopping the simplification process with the help of a geometric distortion criterion. The role of the Gaussian curvature-driven mesh flattening phase, which makes it possible to directly project the simplified model onto the unit sphere, is here fundamental.

IV.4. CONCLUSIONS

In this chapter, we first proposed a survey of the most representative 3D mesh parameterization techniques. The analysis of the state of the art showed that determining a smooth and valid parameterization for 3D triangular meshes still remains a challenging task, especially when certain distortion measures (in terms of angles, lengths, areas) need to be controlled or minimized. Thus, the main challenges of any parameterization concern the guarantee of no triangle overlappings and of low distortions.

While meshes with disk topology are naturally mapped in a planar domain; closed, manifold, genus-0 meshes are topologically equivalent to a sphere and hence the most natural parameter domain for them is the unit sphere. Both types of approaches are presented and discussed, with principles, advantages and limitations. In addition, some parameterization techniques dedicated to more complex models of arbitrary genus have also been presented.

Two main contributions have been introduced in this chapter. The first one concerns a planar parameterization technique, so-called edge length ratio preserving (ELRP) parameterization. The method involves a barycentric technique based on length ratio preservation. The experimental results demonstrate the superiority of our method compared with other state of the art algorithms by providing low distortions rates in terms of area and lengths, especially for complex objects, with distortion reduction of more than 78,5% and 57% respectively.

The second method concerns a novel spherical parameterization method based on a Gaussian curvature criterion. The proposed approach makes it possible to detect iteratively salient mesh vertices and to locally flatten them, until a sphere-like surface is obtained,

adapted to a direct spherical mapping. The experimental evaluation, carried out on a set of 3D models of various shapes and complexities, shows that the proposed method makes it possible to reduce both angle and area distortions with more than 35% and 19% respectively.

Finally, as a key factor of the proposed method, let us mention its complete automatic nature: our planar and spherical parameterization algorithms do not require any human intervention.

V. MESH DEFORMATION FOR FEATURE ALIGNMENT

Summary: *This chapter tackles the issue of feature alignment between the source and target models considered in the morphing process. We solve this problem in the parametric domain with the help of various mesh warping techniques. However, not all existing deformation techniques are well-suited for our purpose. Thus, in this chapter, we propose an evaluation of the warping algorithms and we retain the ones that meet the constraints related to feature alignment of meshes defined in the parametric domain and which lead to a minimum mesh distortion.*

V.1. INTRODUCTION

After the parameterization of the two source and target models, we can directly overlay the obtained embeddings, apply an arbitrary interpolation procedure and obtain a morphing sequence. However, such an approach would fail keeping aligned the relevant characteristics of the 3D models to be morphed in the intermediate morphing models, and would suffer from the same limitations as the simple cross-dissolve techniques discussed in Chapter III.

The characteristics of the 3D models are described by a set of features, specified on both source and target models and supposed to be available. In a general manner, such features are defined as sets of points, lines, curves on the corresponding 3D surfaces. Figure V.1 illustrates such a set of features, for a 3D model of a face. Here, the features correspond to the eyes, mouth, nose, ears, forehead.

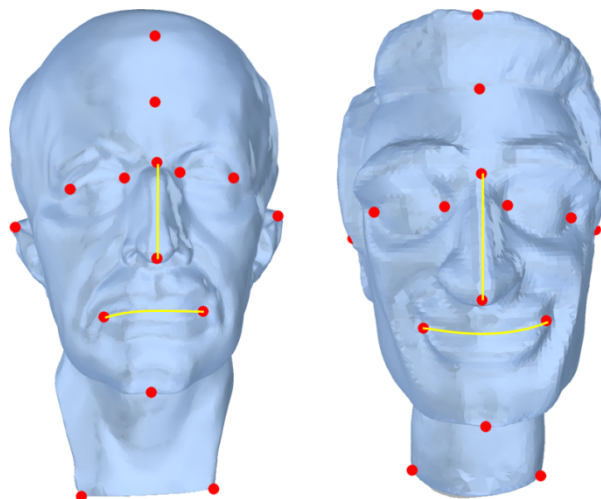


Figure V.1. 3D Mesh models and associated feature points.

In our work, we considered uniquely sets of feature points, defined as vertices of the source and target meshes. Each point on the source model has its correspondent point on the target object.

Let us note that such features are strongly dependent of each 3D representation. Even if some automatic feature extraction and matching methods are available in some particular cases, the general case requires a manual specification.

In order to ensure that the features are preserved during the morphing process, the corresponding feature points should have the same positions in the parametric domain (vertex to vertex correspondence). However, this property is not guaranteed by any mesh

parameterization method and the feature points can have strongly different positions in the parametric domain (Figure V.2).

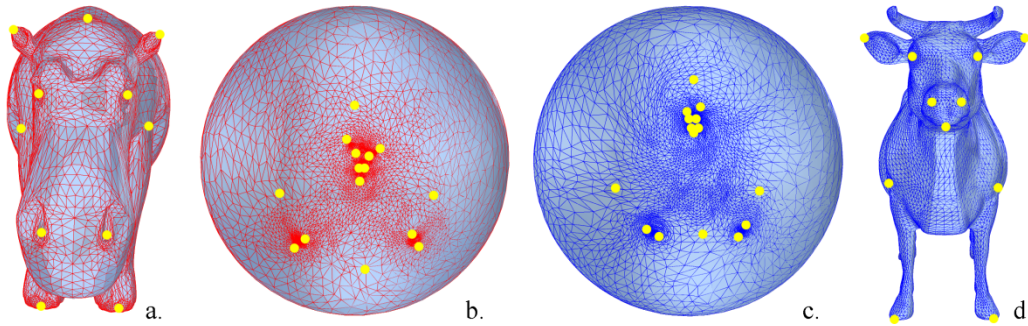


Figure V.2. Illustration of two parameterizations where feature are not aligned: (a), (d) original models; and (b), (c) their embeddings.

Thus, in a first phase, it is necessary to re-place the corresponding feature points such that they share the same position in the parameter domain. Such a re-placement requires a global deformation of the whole parametric domain, such that the corresponding 2D maps should be smoothly deformed without foldovers. Such a process is referred to as *mesh warping*. In order to accomplish this task, it is necessary to consider appropriate mesh deformation techniques.

Various shape deformation methods have been developed within the context of various applications (e.g., 3D animation techniques, special effects, viseme synthesis...). Let us analyze further how the deformation techniques presented in the literature can suit the mesh warping purposes.

V.2. RELATED WORK

V.2.1. Space deformations

With space deformations, a deformed shape is obtained by repeated transformations of the space in which the initial shape is embedded. In 3D, a space deformation can be defined by a global function $U: \mathbb{R}^3 \rightarrow \mathbb{R}^3$, where:

$$U(p) = U(p_1, p_2, p_3) = \begin{bmatrix} U_1(p_1, p_2, p_3) \\ U_2(p_1, p_2, p_3) \\ U_3(p_1, p_2, p_3) \end{bmatrix} \quad (\text{V.1})$$

Historically, the first global function used as a modeling tool was introduced by Barr in [Bar84]. Barr refers to U as a globally specified deformation and proposes several examples including functions for twisting, bending and tapering. Such deformations are still used today and are incorporated into various modeling and animation software as so-called nonlinear deformer [Ali05]. Figure V.3 illustrates some examples of space deformations.

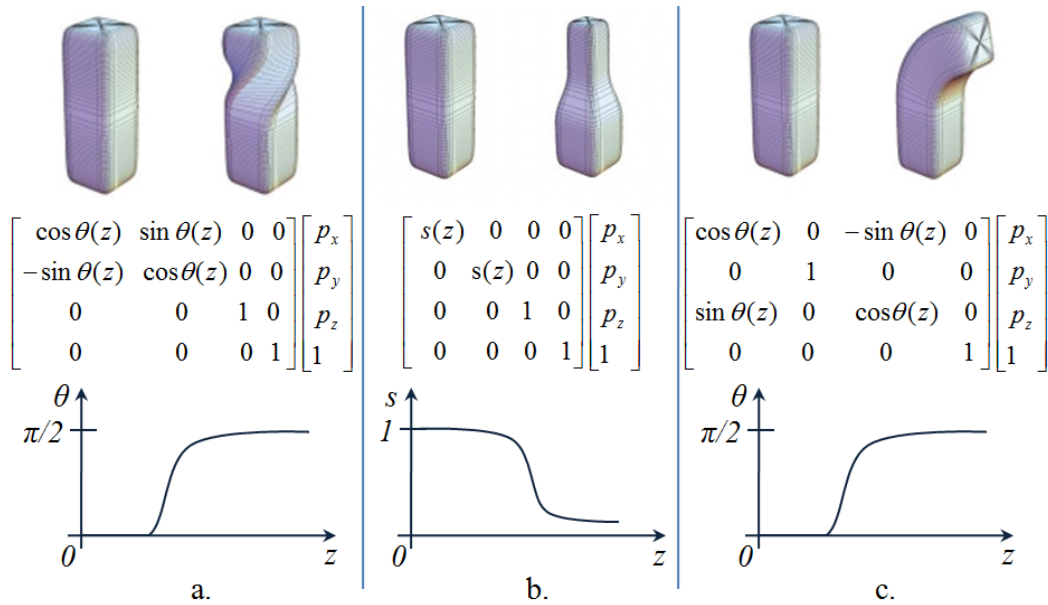


Figure V.3. Space deformations [Bar84]: (a) rotation in z – twist; (b) scale – taper; (c) rotation in y – bend.

Barr also defines a locally specified deformation as the 3x3 Jacobian matrix of U :

$$J = \frac{\partial U}{\partial p} = \begin{bmatrix} \frac{\partial U_1}{\partial p_1} & \frac{\partial U_1}{\partial p_2} & \frac{\partial U_1}{\partial p_3} \\ \frac{\partial U_2}{\partial p_1} & \frac{\partial U_2}{\partial p_2} & \frac{\partial U_2}{\partial p_3} \\ \frac{\partial U_3}{\partial p_1} & \frac{\partial U_3}{\partial p_2} & \frac{\partial U_3}{\partial p_3} \end{bmatrix} \quad (\text{V.2})$$

The matrix J indicates how differential vectors are transformed by the function U . In addition, a method to convert from a locally specified deformation of a primitive back to a global specification via integration is also proposed. Starting from an arbitrary origin (the constant of integration), the differential changes are integrated across the primitive to determine the globally deformed positions.

In his pioneering work, Barr set the premises of the well-known free-form deformations (FFD), recalled in the next section.

V.2.2. Free-form deformations

Free-form deformations (FFD) represents a space deformation technique originally formulated by Sederberg and Parry [Sed86] and then extended by MacCracken [Mac96] or, more recently, by Ju *et al.* [Ju05].

The FFD principle consists of embedding the 3D model to be deformed into a 3D lattice of control points. Such a set of control points makes it possible to define a global deformation of the \mathbb{R}^3 space, by considering for example B-Spline or NURBS functions.

A useful property of free-form deformations is that the generated transformation is independent of the complexity of the model being deformed. Another advantage comes from manipulation capabilities of such a deformation: the user can locally and intuitively control the deformation by modifying the position of the desired control points.

Figure V.4 shows an example of such deformation. The deformation complexity is correlated with the density of the control lattice. FFDs only need a very coarse regular control lattice to create coarse-scale deformations of a model. However, for finer-scale deformations, a very dense control lattice is usually required.

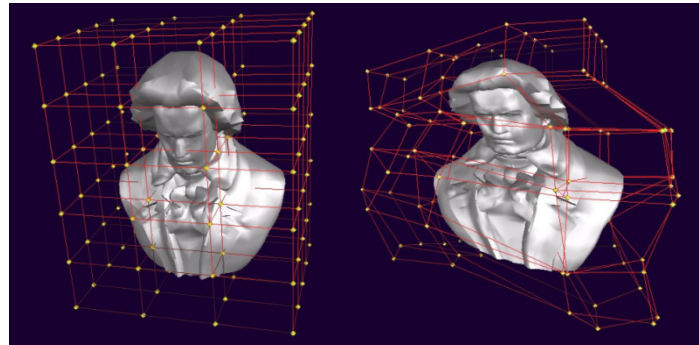


Figure V.4. Free form deformation.

Since the mapping from the lattice to the model is generally defined without considering the embedded model geometry, FFD may incorrectly influence regions that are spatially close with respect to a Euclidian distance, but relatively far in what concerns the geodesic distance. In addition, the lattice-based approaches have a low precision in moving vertices.

FFDs are often used in professional modeling applications (e.g., 3DS Max, Maya), as they are computationally fast and do not require any constraints regarding the representations of the models (e.g., irregular meshes, point clouds, parametric surfaces).

Borrowing the principle of defining a deformation with a set of controllers, Singh [Sin98] proposes to use domain curves, so-called *wires*, to define the domain of deformation for an object (Figure V.5). Wires follow the deformable features of an object as such they provide a coarse geometric representation of the model, together with an intuitive way to deform it.

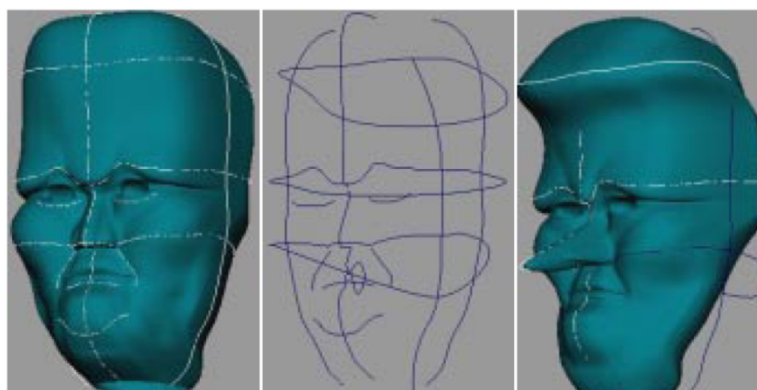


Figure V.5. Wires: A geometric deformation technique [Sin98].

Sumner *et al.* [Sun07] propose to use the so-called deformation graph, which is a more general deformation domain, for intuitive deformation of a wide range of shape representations and editing scenarios. Their method supports direct manipulation of a mesh being deformed and makes the deformation graph transparent to the user.

FFD methods are suitable for smooth surfaces, but present several drawbacks when dealing with objects with a high level of details (such as those acquired from scanning devices). Since the deformations are globally controlled by the lattice grid, the details of the shape cannot be preserved in an efficient manner.

V.2.3. Skeletal deformation

Skeletal deformations (Figure V.6) are highly popular in the field of real-time animation of articulated 3D models. They can also be applied to a wide range of soft objects, for example to cloth simulation [Cor05].



Figure V.6. Skeleton based deformation [Yan08b].

Such techniques exploit a hierarchical structure of object's skeleton. The skeleton is represented as a tree structure whose nodes are identified with the joints (given by their positions and orientations) and edges with the corresponding bones. A skeleton provides the domain of deformation for the 3D mesh. For each bone, a region of influence (*i.e.*, set of vertices) on the 3D mesh is associated to. Thus, when moving the skeleton's bones, the associated skin, *i.e.*, the 3D model surface, will be displaced accordingly. A linear weighting method is applied at the level of joints in order to avoid foldovers at the level of surface points that are influenced by multiple bones. The vertex weights, which denote the amount of influence of individual bones, must be specified during the so-called *skinning* process. The deformation of each vertex is then defined as a weighted blending of the transformations of its associated bones. Let us note that the quality of the deformation is strongly influenced by this weighting mechanism.

Automatic skeleton extraction according to the geometric information of a given mesh is in general very difficult. The topology of the extracted skeleton is often not satisfactory since the

extraction process is sensitive to the shape perturbation of the skin surface. Instead, Baran and Popovic [Bar07] present a method to automatically fit a given template skeleton with the fixed topology to a mesh.

Skeletal deformation is particularly useful for objectives of virtual character animation, where the hierarchical skeleton structure fits well the anatomy of the considered characters. However, in a more general case and notably for mesh warping purposes, defining appropriate sets of bones/joints is not straightforward.

In this case, more general deformation techniques have to be considered. A first solution is provided by the so-called multiresolution mesh editing methods, described in the next section.

V.2.4. Multiresolution mesh editing

One of the very first multiresolution shape editing was introduced [Zor97]. The underlying principle of multiresolution mesh editing consists of hierarchically decomposing a complex object into a coarse, base mesh and a set of gradually finer levels of detail. The differences between each level of detail are stored in the associated representation, for reconstruction purposes.

Analogously to Fourier analysis, this process can be interpreted as a decomposition of the 3D geometry signal into low and high frequencies. Let us note that a generalization of the wavelet transform to 3D mesh models can be obtained with the help of such a representation.

Zorin *et al.* [Zor97] combine this technique with a free-form deformation in order to achieve a detail preserving mesh editing tool. The manipulation is done in the classic FFD manner, but the user is allowed to select a specific level of detail. If only the base mesh is deformed, all the details, corresponding to the higher frequency components of the mesh are retained. The advantage, with respect to the classic FFD, is the detail preserving editing that allows the manipulation of complex shapes with a large number of vertices.

The drawback of the method relates to the disturbing artifacts that might appear at the borders between patches of the base mesh. Here, different deformations are applied independently to each patch, which does not guarantee the creation of a globally smooth deformation field.

A solution to this problem is proposed by the so-called Laplacian mesh editing technique, described in the following section.

V.2.5. Laplacian mesh editing

The mesh deformation approaches discussed so far directly apply transformation to a Cartesian representation of the 3D model's geometry. Since an important goal to achieve when considering mesh deformation concerns the detail preservation, it would be more advantageous to consider an intrinsic, differential mesh representation, where such details can be identified and preserved.

Differential representations can capture information about the local shape properties of a mesh, such as curvature, scale or orientation. One of the most popular differential representation of a 3D geometry concerns the so-called Laplacian coordinates (also known as differential coordinates or δ -coordinates). Laplacian coordinates have been first used for 3D mesh morphing and deformation purposed in [Ale03].

Let us recall the definition of Laplacian coordinates. Let us consider a mesh $M(V, E, F)$ with V, E and F respectively denoting the sets of vertices, edges and faces (triangles). For each mesh vertex $p_i(x_i, y_i, z_i)$, the differential (or the δ -coordinates) are defined as the difference between the absolute coordinates of p_i and the center of mass of its adjacent vertices:

$$\delta_i = \delta_i \left(\delta_i^{(x)}, \delta_i^{(y)}, \delta_i^{(z)} \right) = p_i - \frac{1}{d_i} \sum_{j \in N(i)} p_j \quad , \quad (\text{V.3})$$

where $N(i) = \{j | (i, j) \in E\}$ and $d_i = |N(i)|$ is the number of vertices adjacent to p_i (i.e, its valence). Globalizing this transformation to the whole mesh can be written in matrix form. Let us consider A the adjacency (connectivity) matrix of the mesh, defined as:

$$A_{ij} = \begin{cases} 1 & , \text{if } (i, j) \in E \\ 0 & , \text{otherwise} \end{cases} \quad (\text{V.4})$$

Let us also denote by D the $(V \times V)$ diagonal matrix such that $D_{ii} = d_i$. Then, the Laplacian matrix L is defined as:

$$L = I - D^{-1}A \quad (\text{V.5})$$

By applying the linear operator associated to the Laplacian matrix L to the geometry signal, we obtain the following equation, which describes the relation between Cartesian and Laplacian coordinates:

$$L_S X = D \delta^{(x)}, L_S Y = D \delta^{(y)}, \text{ and } L_S Z = D \delta^{(z)}, \quad (\text{V.6})$$

where X (resp. Y and Z) is an V -dimensional vector containing the x (resp. y and z) Cartesian values of all the mesh vertices, and $\delta^{(x)}$, $\delta^{(y)}$ and $\delta^{(z)}$ are the corresponding Laplacian coordinates.

In practice, it is more convenient to consider the symmetric version of the L matrix defined as:

$$L_S = DL = D - A \quad , \quad (V.7)$$

where

$$(L_S)_{ij} = \begin{cases} d_i & i = j \\ -1 & (i,j) \in E \\ 0 & \text{otherwise} \end{cases} . \quad (V.8)$$

Figure V.7 presents an example of a mesh and its associated matrices. The matrix L_S (or L) is called the topological (or graph) Laplacian of the mesh. Graph Laplacians have been extensively studied in algebra and graph theory [Chu97], primarily because their algebraic properties related to the combinatorial aspects of the graphs they represent.

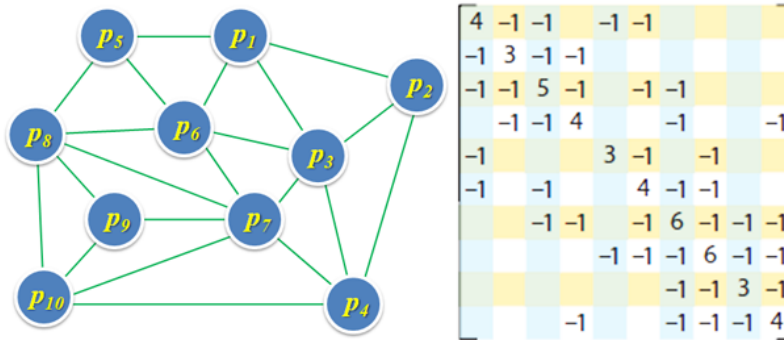


Figure V.7. An example of a triangular mesh and its associated symmetric Laplacian matrix.

Let us observe that equation (V.3), expressing the differential coordinates of a vertex p_i , can be re-written as:

$$\delta_i = \frac{1}{d_i} \sum_{j \in N(i)} (p_i - p_j) \quad (V.9)$$

If we now assume that the mesh M represents a piecewise-linear approximation of a smooth surface, then the sum in equation (V.8) can be interpreted as a discretization of the following curvilinear integral:

$$\frac{1}{|\gamma|} \int_{p \in \gamma} (p_i - p) dl(p) \quad , \quad (V.10)$$

where γ represent a closed surface curve around vertex p_i and $|\gamma|$ is the length of γ .

It is known from differential geometry [Tau95] that:

$$\lim_{|\gamma| \rightarrow 0} \frac{1}{|\gamma|} \int_{p \in \gamma} (p_i - p) dl(p) = -H(p_i) n_i \quad , \quad (V.11)$$

where $H(p_i)$ and n_i respectively denote the mean curvature and the normal vector at point p_i .

The orientation of the differential coordinate vector δ_i approximates the local normal direction and its magnitude is proportional to the local mean curvature. Thus, δ_i encapsulates the local shape information of the considered surface.

However, the above-described discrete approximation does not hold in the case of a non-uniform, irregular mesh sampling. In order to overcome such a drawback, equation (V.9) can be extended by considering a weighting scheme as described in the following equation:

$$\delta_i = \frac{1}{\sum w_{ij}} \sum_{j \in N(i)} w_{ij} (p_i - p_j) \quad (\text{V.12})$$

where w_{ij} is the weight associated to the edge (i, j) .

There are several alternatives to define the weights w_{ij} . When $w_{ij} = 1 \forall i, j$, then equations (V.9) and (V.12) become identical. This scheme is called *uniform weighting*. It only describes the topological properties of the mesh, but not the geometrical ones, since the coordinates δ are defined by the mean of the surrounding vertices without considering their geometry.

Two other different weighting schemes are proposed in the literature, so-called the cotangent weighting and tangent weighting. They are inspired from the [Eck95] and [Flo03] parameterization methods presented in Chapter IV.

Thus, Meyer *et al.* [Mey03] propose to use the so called cotangent weights, defined as:

$$\delta_i^c = \frac{1}{|\Omega_i|} \sum_{j \in N(i)} \frac{1}{2} (\cot \alpha_{ij} + \cot \beta_{ij}) (p_i - p_j) \quad (\text{V.13})$$

where $|\Omega_i|$ is the size of the Voronoi cell of i and α_{ij}, β_{ij} represent the two opposite angles of edge (i, j) .

However, since the cotangent weights can be negative due to large angles in the mesh structure, usually it is more convenient to use the mean value coordinates, also called tangent weights, defined as described in Section IV.2.2.1.

The reconstruction of the mesh surface (*i.e.*, recovery of the initial geometry in Cartesian coordinates from differential coordinates) can be obtained by solving the following linear system of equations:

$$L\mathbf{P} = \delta \quad (\text{V.14})$$

for each dimension X, Y and Z .

Let us observe that the matrix L is singular, since its rows sum up to zero. More precisely, it can be shown that matrix L has the rank $V - 1$ if the mesh is connected [Sor06]. This property is related to the translational invariance property of the differential coordinates. As a consequence, matrix L is not invertible and therefore, the system of equations (V.14) is not analytically solvable. Instead, it should be solved in the mean square error sense, for example with the help of a pseudo-inverse method [Pen55].

In order to obtain a unique solution is necessary to specify the Cartesian coordinates for at least one mesh vertex. In practice, such coordinates are specified more generally for a set of vertices whose spatial location is known. Such points are called constraint (or anchor) points. The system (V.14) becomes in this case:

$$\begin{pmatrix} L \\ \omega I_{m \times m} | 0 \end{pmatrix} \mathbf{P} = \begin{pmatrix} \delta^P \\ \omega c_{1:m} \end{pmatrix}, \quad (\text{V.15})$$

where c denotes a constraint on spatial location and ω represent a weight that can be used to influence the importance of the positional constraints.

Let us observe that solving equation (V.15) in the mean square error sense is equivalent to minimizing the following energy functional:

$$\tilde{x} = \underset{x}{\operatorname{argmin}} (\|Lx - \delta^{(x)}\|^2 + \sum_{j \in c} \omega^2 |x_j - c_j|^2) \quad (\text{V.16})$$

The system in equation (V.15) is a sparse linear system that can be efficiently solved with dedicated representations. Thus, a general storage scheme is the so-called *compressed column storage* format [Pre02]. Here, only three vectors are used to store all the Laplacian matrix values:

- a first one for the nonzero values as they are traversed column by column,
- a second vector for the corresponding row indices of each value, and
- a third vector to store the locations in the other two arrays that start a column.

Sparse matrices provide the possibility to significantly accelerate the classic matrix algorithms. A well known efficient algorithm consist in the Cholesky decomposition. The solution of the linear system is precomputed with a Cholesky factorization that splits the matrix in an upper and a lower triangular matrix. This decomposition is done once, while the new coordinates are computed very fast by a simple forward and back substitution for each dimension.

The classical Laplacian coordinates method as presented above solves equation (V.15) in the sense of least squares minimization. This leads to low displacement accuracy (*i.e.*, the anchor vertices will not reach the exact final position established). Thus, we propose to modify the system by replacing the normal equations corresponding to control vertices with constraint equations that impose those vertices to reach their correct final position.

In the case of unitary weights, we will call this technique the Uniform Fix Laplacian coordinate deformation method (UFLC). For mean value coordinates weights, we will call it the Tangential Laplacian coordinate deformation method (TLC).

Another high popular approach for 3D mesh deformation is based on the so-called radial basis functions.

V.2.6. Radial basis functions

The radial basis function (RBF) approach represents an important tool in approximation theory due to spectral accuracy, flexibility with respect to geometry, dimensional independence and ease of implementation especially when interpolating scattered data in multidimensional spaces.

In the case of mesh deformation, the RBF approach makes it possible to interpolate the displacement of the whole set of mesh vertices based only on the known displacement of some control points. The method offers the advantage that no grid connectivity information is required and only a small system of equation needs to be solved depending on the number of constrained vertices. The displacement of the mesh vertices are characterized by an interpolation function s , which is defined as the sum of a set of radial basis functions, as described in the following equation:

$$\forall j = 1, \dots, V, \quad s(p_j) = \sum_{i=1}^{n_c} \alpha_i \phi(\|p_j - p_{c_i}\|) + P(p_j), \quad (\text{V.17})$$

where V is the total number of mesh vertices, n_c the number of control points, $p_{c_i} = p_{c_i}(x_{c_i}, y_{c_i}, z_{c_i})$ their spatial positions, P a given polynomial function of degree $|P|$, and ϕ is the given radial basis function defined with respect to the Euclidian distance ($\|\cdot\|$).

The coefficients of the polynomial P and the coefficients $\alpha_i = [\alpha_{ix}, \alpha_{iy}, \alpha_{iz}]$ in equation (V.17) can be determined from the interpolation conditions ((V.18) and (V.19)):

$$\forall j = 1, \dots, n_c, \quad s(p_{c_j}) = d_{c_j} \quad (\text{V.18})$$

where d_{c_i} is a vector specifying the displacement of the control vertex p_{c_i} .

When the polynomial term P is included, the system is completed with the additional conditions:

$$\sum_{i=1}^{n_c} \alpha_i q(x_{c_i}) = 0 \quad (\text{V.19})$$

for all polynomials q with a degree equal to or less than $|P|$.

According to Boer *et al.* [Boe07], the minimal degree of P depends on the choice of the basis function ϕ . More precisely, an unique interpolant is needed if the basis functions are positively defined. If the basis functions meet this requirement and they are of order less than or equal to 2, then a linear polynomial can be used. Since linear polynomials have the

property to recover exactly a model after a rigid transformation, we only further consider solely basis functions that satisfy this criterion.

The values for α_j and the polynomial P are determined by rewriting equation (V.17) as follows:

$$\begin{bmatrix} d_c \\ 0 \end{bmatrix} = \begin{bmatrix} M_{c,c} & R_c \\ R_c^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad (\text{V.20})$$

with α containing the coefficients α_i , β the coefficients of the linear polynomial P , R_c an $n_c \times 4$ matrix where each row i is given by $[1 \ x_{c_i} \ y_{c_i} \ z_{c_i}]$ and $M_{c,c}$ an $n_c \times n_c$ matrix containing the basis function:

$$M_{c,c} = \begin{bmatrix} \phi_{s_1 s_1} & \phi_{s_1 s_2} & \cdots & \phi_{s_1 s_{n_c}} \\ \vdots & \vdots & & \vdots \\ \phi_{s_{n_c} s_1} & \phi_{s_{n_c} s_2} & \cdots & \phi_{s_{n_c} s_{n_c}} \end{bmatrix} \quad (\text{V.21})$$

with $\phi_{c_i c_j} = \phi(\|p_{c_i} - p_{c_j}\|)$.

Once the coefficients α and β have been determined, the interpolation function in equation (V.17) can be used to compute, point by point, the displacement of all non-constraint vertices of the mesh.

Let us note that the determined displacement is interpolated separately for each spatial direction. Also, the size of the system that has to be solved in (V.20) is equal to $(n_c + 4) \times (n_c + 4)$, which is relatively small, depending on the number of specified control vertices.

In order to avoid numerical stability issues the linear system of equations (V.20) is solved with the help of singular value decomposition (SVD)-based pseudo-inverse method [Pre02], ensuring a least square solution.

Let us note that in this case, only an approximate solution can be obtained, *i.e.*, the actual deformations $s(p_{c_i})$ obtained for the control points will approximate, in the means square error sense the specified displacements d_{c_i} .

There are various radial basis functions available in the literature, which can be divided in two groups: functions with compact support and functions with global support. Functions with compact support can be defined as:

$$\phi(x) = \begin{cases} f(x) & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases} \quad (\text{V.22})$$

where $f(x) \geq 0$. The function is generally scaled with a support radius r to control the compact support. The following composite function is then obtained:

$$\phi_r = \phi(x/r) = \phi(h) \quad (\text{V.23})$$

In this manner, solely the mesh vertices lying inside a sphere of radius r around the control point p_i are influenced by the movement of this point.

Thus, higher values for r lead to more global deformation solutions, but with the cost of dense matrix systems that might augment the computational time needed to solve equation (V.21).

On the contrary, lower values of r result in more local deformation fields, concentrated around the considered control point. In addition, this will yield a sparse matrix in equation (V.21) which can be solved more efficiently.

Table V.1 summarizes various radial basis functions reported in the literature, with either compact or global support.

Table V.1. Radial basis functions.

No.	Name	Type	Radial basis function
1.	CP C^0	CS	$(1 - h)^2$
2.	CP C^2	CS	$(1 - h)^4(4h + 1)$
3.	CP C^4	CS	$(1 - h)^6\left(\frac{35h^2}{3} + 6h + 1\right)$
4.	CP C^6	CS	$(1 - h)^8(32h^3 + 25h^2 + 8h + 1)$
5.	CTPS C^0	CS	$(1 - h)^5$
6.	CTPS C^1	CS	$1 + \frac{80h^2}{3} - 40h^3 + 15h^4 - \frac{8h^5}{3} + 20h^2 \log_2(h)$
7.	CTPS C^2_a	CS	$1 - 30h^2 - 10h^3 + 45h^4 - 6h^5 - 60h^3 \log_2(h)$
8.	CTPS C^2_b	CS	$1 - 20h^2 + 80h^3 - 45h^4 - 16h^5 + 60h^4 \log_2(h)$
9.	Thin plate spline (TPS)	GS	$x^2 \log_2(x)$
10.	Multiquadric biharmonics (MQB)	GS	$\sqrt{a^2 + x^2}$
11.	Inverse multiquadric biharmonics (IMQB)	GS	$\sqrt{\frac{1}{a^2 + x^2}}$
12.	Quadric biharmonics	GS	$1 + x^2$
13.	Inverse quadric biharmonics	GS	$\frac{1}{1+x^2}$
14.	Gaussian	GS	e^{-x^2}

CS – Compact support

GS – Global support

The compact support property of the RBFs in rows 1 to 8 in Table V.1 is ensured by truncating the function to the $[0, 1]$ interval. Let us note that this is always possible, the resulting functions remaining continuous since they take the value 0 in the set $\{0, 1\}$.

The functions $CP C^0$, $CP C^2$, $CP C^4$ and $CP C^6$ are based on polynomials chosen to have the lowest degree of all polynomials that create a C^n continuous basis function with $n \in \{0, 2, 4, 6\}$. The next four are a series of functions based on the thin plate spline interpolation which creates C^n continuous basis functions with $n \in \{0, 1, 2\}$.

The MQB and IMQB techniques use the a parameter in order to control the shape of the basis functions. A large value of a return flat functions, while small values of a gives narrow, localized functions.

In order to establish an objective comparison between various deformation methods, it is necessary to first specify a set of mesh quality metrics. The adopted solutions are described in the following section.

V.3. MESH QUALITY METRICS

The considered mesh quality metric is based on the approach proposed by Knupp [Knu03] which uses a set of Jacobian matrices defined for each mesh triangle. The method is dedicated to 2D meshes, such as those obtained after a parameterization process.

Considering a mesh triangle, with the coordinates of the vertices defined by (x_k, y_k) , $k = 1, 2, 3$ where k denotes the vertices of the triangle, we can construct three Jacobian matrices A_k , one around each node k as:

$$\forall k \in \{1, 2, 3\}, \quad A_k = \begin{bmatrix} x_{k+1} - x_k & x_{k+2} - x_k \\ y_{k+1} - y_k & y_{k+2} - y_k \end{bmatrix} \quad (\text{V.24})$$

Since the determinant α_k of each Jacobian matrix in equation (V.24) represents twice the area of the considered triangle and it is independent of the node on which it is computed, the subscript k can be skipped.

Additionally, a metric tensor λ can be computed as:

$$\lambda = A^T A \quad (\text{V.25})$$

Matrix λ is a 2x2 symmetric matrix with components λ_{ij} , $i, j = 1, 2$. Intuitively, λ_{11} and λ_{22} are measures of the squared lengths of two triangles edges and λ_{12} is a measure of the angle between them. Thus, let us also note that the dot product between the two edges is given by:

$$\lambda_{12} = \sqrt{\lambda_{11} \lambda_{22}} \cos \theta \quad (\text{V.26})$$

where θ is the angle between the two sides joined at the considered node.

It can be shown that the triangle area a can be expressed as:

$$a^2 = \lambda_{11} \lambda_{22} - \lambda_{12}^2 = \lambda_{11} \lambda_{22} \sin^2 \theta \quad (\text{V.27})$$

The size metric f_{size} is defined as:

$$f_{size} = \min\left(\tau, 1/\tau\right) \quad (\text{V.28})$$

where $\tau = a/w$ is the ratio between the area of a triangle in the deformed mesh and the area of the reference (initial) triangle.

The measure f_{size} reaches its maximum value, equal to 1, if and only if the final mesh triangle has the same area as the reference triangle. On the contrary, when f_{size} is equal to zero, the deformed triangle is degenerated (*i.e.*, zero area).

The shape quality metric f_{shape} aims at measuring distortions in the shape of triangle, independently of its size, and is defined relatively to an equilateral triangle as described by the following equation:

$$f_{shape} = \frac{\sqrt{3}\alpha}{\lambda_{11} + \lambda_{22} - \lambda_{12}} \quad (V.29)$$

Using equation (V.27) and the law of cosines, equation (V.29), can be rewritten in a form which shows the relationship of the shape quality metric and the angle at the considered point:

$$f_{shape} = \frac{\sqrt{3}r \sin\theta}{1 - r \cos\theta + r^2} \quad (V.30)$$

where $r = \sqrt{\lambda_{22}/\lambda_{11}}$.

The shape quality metric f_{shape} is equal to 1 if the final mesh triangle is equilateral, and it is zero if the triangle is degenerate.

Finally, for each triangle of the mesh, the quality metric is a scalar quantity defined by the product:

$$f_{ss} = f_{size} \cdot f_{shape} \quad (V.31)$$

The triangle with $f_{ss} = 1$ refers to the ideal triangle (*i.e.*, the equilateral triangle with the area equal to the area of a pre-established triangle considered as the reference/initial element).

The extension of the f_{ss} quality metric for the 3D case is straightforward, taking into account that the r parameter in equation (V.30) is actually the ratio of two consecutive edge lengths, and θ represents the angle between them.

Beside the two metrics presented above, we also seek to evaluate the displacement accuracy of each deformation technique, *i.e.*, the average error between the actual displacement and the specified positions of the control points. Mathematically, we can define a new metric as:

$$Dis = \frac{1}{n_c} \sum_{i=0}^{n_c} \frac{\|p_{c_i} - p_{c_i}^{dis}\|}{\|p_{c_i} - p_{c_i}^{init}\|} * 100 \quad (V.32)$$

where n_c denotes the total number of control vertices, $p_{c_i}^{init}$ and p_{c_i} represent the initial position and the final place the point i should reach, while $p_{c_i}^{dis}$ is the actual position where point i is placed after the movement. Ideally the value of Dis should be zero.

V.4. EXPERIMENTAL EVALUATION

In order to establish the most suitable method that could successfully meet the constraints related to feature alignment of meshes defined in the parametric domain, we have considered the following set of requirements:

- Topology consistency – the mesh warping technique should ensure that the mesh connectivity remains unchanged during the movement.
- No foldovers - the deformation method should not flip the mesh triangles.
- Smoothness – the movement process should be continuous, with no discontinuities.
- Large displacements – the mesh deformation technique should allow relatively large displacements of vertices.
- Accurate displacement – the source mesh feature vertices should reach as close as possible the position of their target counterparts.
- Low distortions – the variation in terms of shape and size of the mesh triangles should be minimized.
- Spatial domain conservation – since we aim to deform meshes in the parametric domains, it is desirable that all mesh vertices remain in the considered domain (*i.e.*, planar or spherical domain).

After analyzing the properties the methods presented in Section V.2. we can conclude that only the Laplacian coordinates and the Radial basis functions techniques seem convenient for mesh warping purposes.

For space, free-form, skeletal or multiresolution based deformation techniques would be too complicated to perform fine locally movements of vertices in the parameterization domain and they would require the user intervention for an accurate displacement.

In order to demonstrate the use of RBF and Laplacian coordinates as mesh deformation strategies we have considered several test cases, including rotation, translation and deformation of a rectangular block specified on a 2D, uniform mesh grid (Figure V.8.a).

In [Boe07], several different radial basis functions were compared for a variety of test cases. The results obtained showed that the CP C^2 function offers the best trade-off between computational efficiency and deformed mesh quality. In our work, we aim at comparing different RBFs and different Laplacian coordinates variants in terms of quality deformation, computational efficiency and moreover, displacement accuracy. Further we establish the most suitable method to warp a parameterized mesh and better maintain model vertices in the parametric domain, without introducing triangle overlapping. We considered three basis functions, namely the CTPS C^2_a , CP C^2 , and Gaussian.

Concerning the methods based on Laplacian coordinates, we aim to analyze the Classical Laplacian coordinates technique that obey to equation (V.15) and solve it in the sense of least squares minimization, as well as the two modified versions described in Section V.2.5. Uniform Fix Laplacian coordinate deformation method (UFLC) and Tangential Laplacian coordinate deformation method (TLC).

V.4.1. Deformation in 2D test cases

The first test scenarios consist of a unit square domain. An inner rectangle is included in this domain and undergoes different translations, rotations and scaling in the 2D space. The unit square domain is sampled uniformly in each direction into 35 points and then triangulated. The resulting mesh is illustrated in Figure V.8.a. The test mesh has a total of 1225 grid vertices of which 136 are boundary nodes that are constrained to remain fixed during the deformation process.

A set of control points, corresponding to a rectangle with initial size of (3×6) intervals of the sampling grid is then defined (Figure V.8.a). Various motions are associated with the test rectangle. More precisely, we have considered the following three different cases:

- the first one concerns a simple translation of the rectangle in the plane, with 6 sampling intervals over both x and y directions (Figure V.8.b),
- the second case involves a translation combined with both a rotation of 45° and with a scaling (with a factor $\sqrt{2}$) (Figure V.8.c),
- the third test considers a high amplitude deformation corresponding to a combined rotation (with 90°), translation (12 intervals along the x direction and 4 interval along the y axis) and scaling (with a factor 2) (Figure V.8.d).

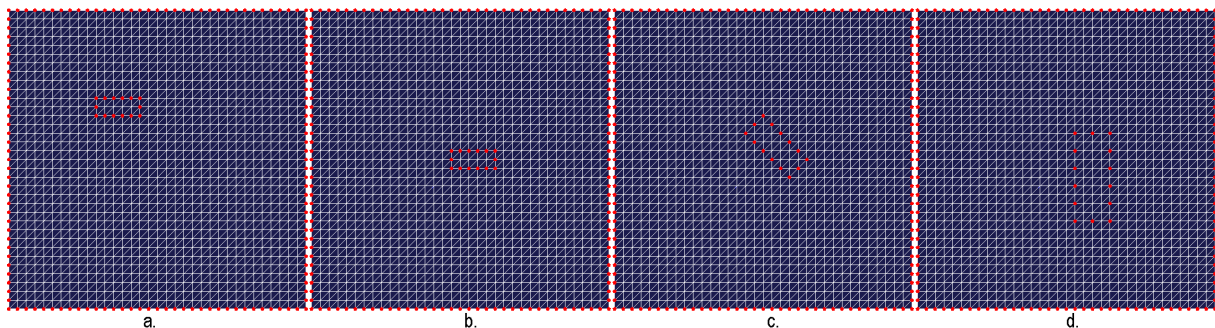


Figure V.8. Test mesh and the control rectangle with the (a) initial position and ((b), (c), (d)) final positions of rectangle corresponding to the three test cases considered.

When using the RBF deformation method, we have retained for evaluation the CTPS C_a^2 and CP C^2 compact basis functions, which were proved to provide the best performances in [Boe07]. The associated support radius (parameter r) has been varied in a range from 2 to 10. In addition, we perform the mesh deformation in a variable number of steps that

iteratively displace the control vertices from their initial position to the final location. The number of intermediary steps ranges from 1 to 50 steps.

In the case of Laplacian techniques, it does not make sense to apply a deformation in steps because for these methods the Laplacian matrix L and the free term δ of the system of equations, are constructed based on the initial shape of the input mesh. If we consider a deformation in steps an intermediary phase n will try to approximate the model obtained at a previous step $n-1$ (and not the original), which translates into an error accumulation that decreases significantly the performance of the algorithm.

Figures V.9 to Figure V.20 present the results obtained for both CTPS C^2_a and CP C^2 RBFs in terms of minimum and average f_{SS} quality metric (respectively denoted by $\min(f_{SS})$ and, $\text{mean}(f_{SS})$), over the whole set of the mesh grid triangles, as function of the number of intermediate steps used to achieve the deformation (from 1 to 50). We also report the values of the displacement accuracy (dis), as well as the computational time for all three proposed scenarios.

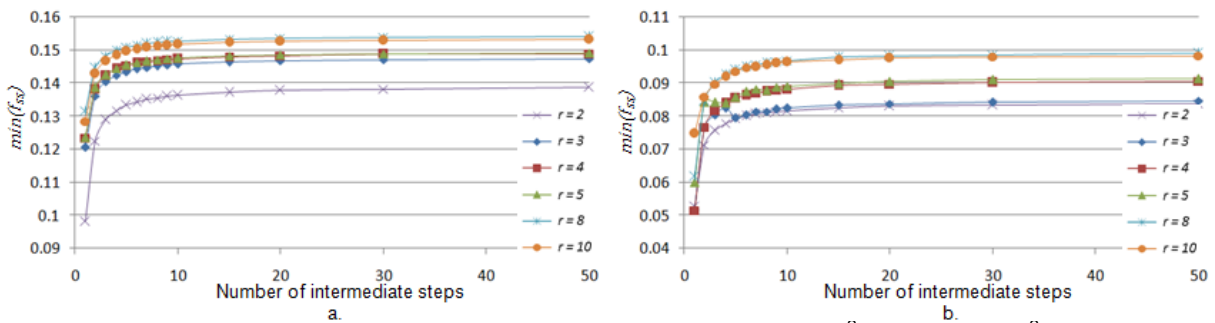


Figure V.9. Quality of the worst triangle of the mesh for (a) CTPS C^2_a and (b) CP C^2 (Case 1).

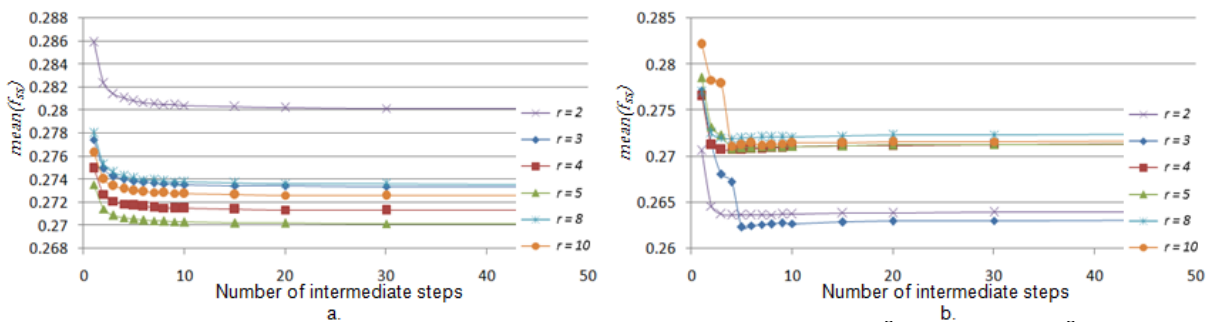


Figure V.10. The mean quality of all triangles in the mesh for (a) CTPS C^2_a and (b) CP C^2 (Case 1)

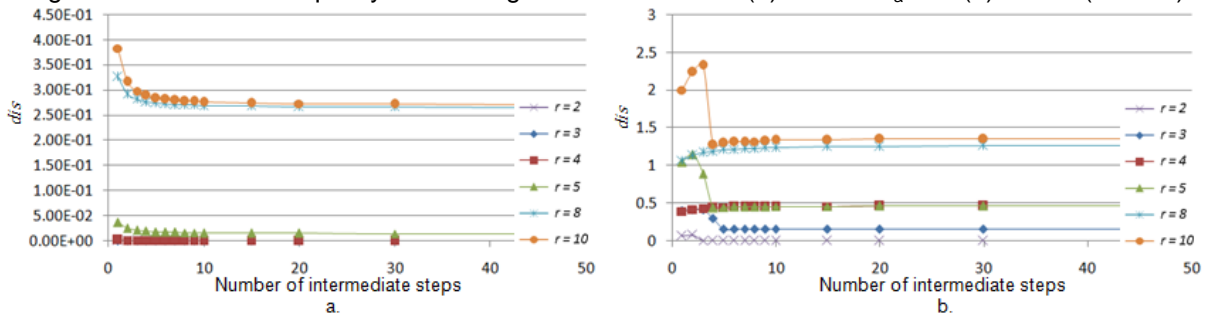


Figure V.11. Accuracy displacement of the control points for (a) CTPS C^2_a and (b) CP C^2 (Case 1).

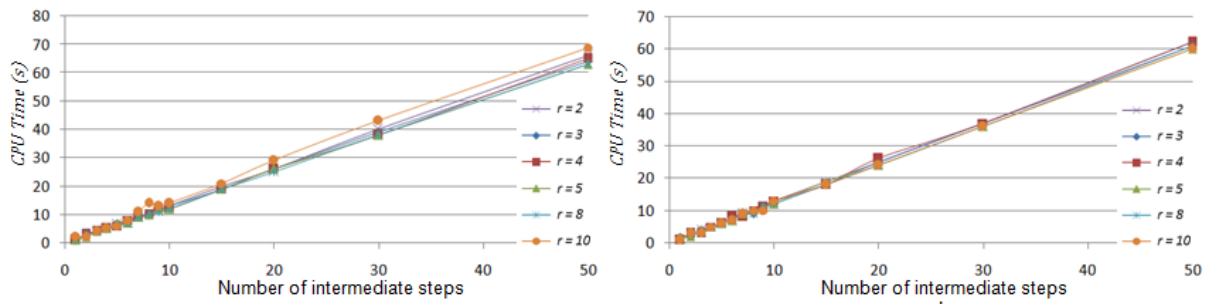


Figure V.12. CPU computational time for (a) CTPS C_a^2 and (b) CP C^2 (Case 1).

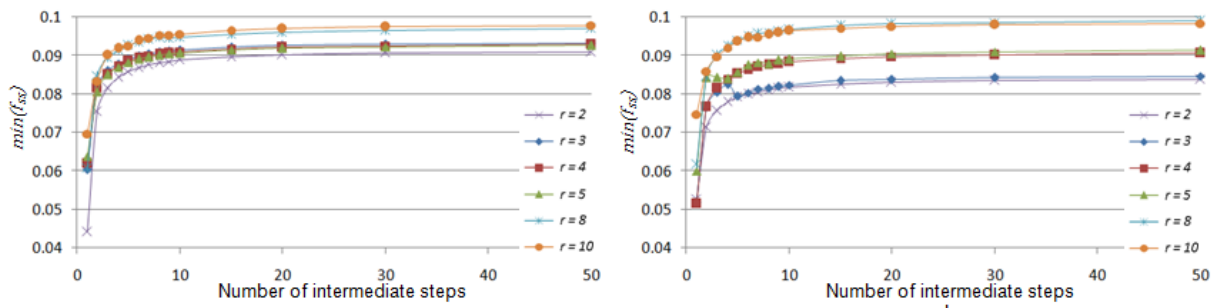


Figure V.13. Quality of the worst triangle of the mesh for (a) CTPS C_a^2 and (b) CP C^2 (Case 2).

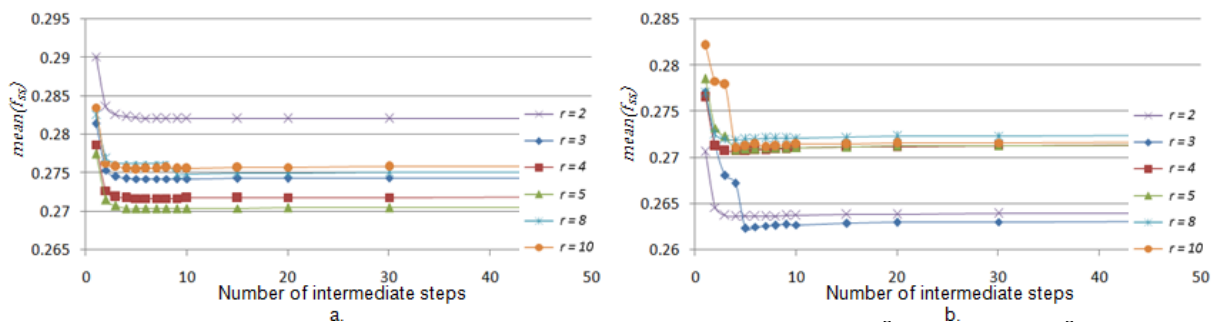


Figure V.14. The mean quality of all triangles in the mesh for (a) CTPS C_a^2 and (b) CP C^2 (Case 2).

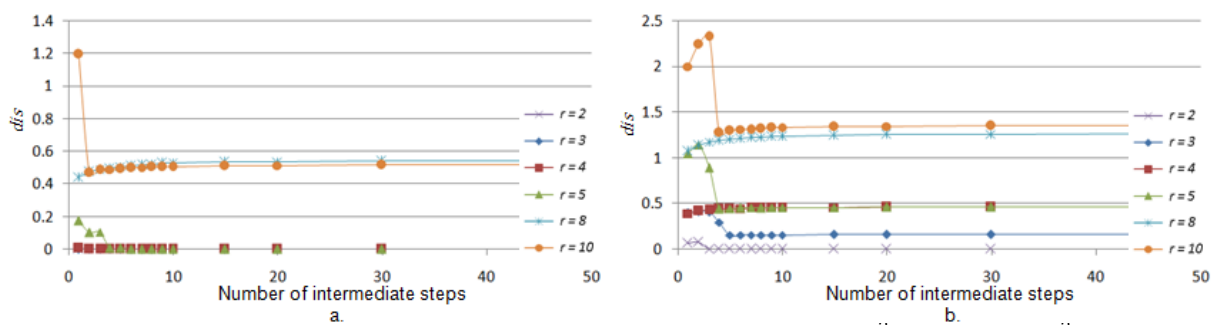


Figure V.15. Accuracy displacement of the control points for (a) CTPS C_a^2 and (b) CP C^2 (Case 2).

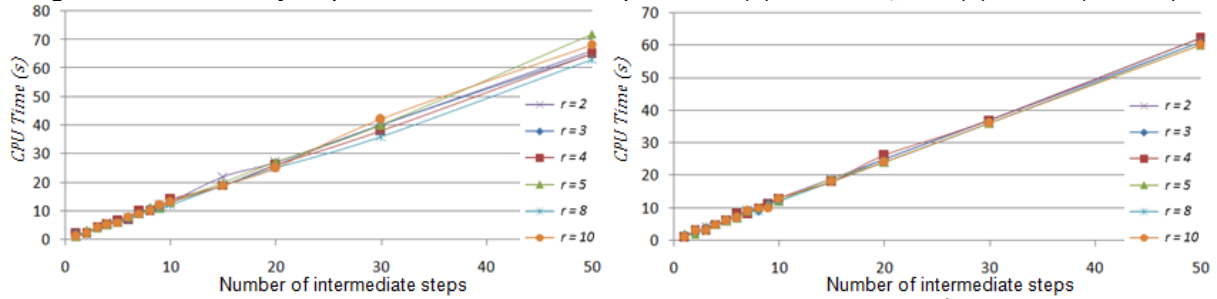


Figure V.16. CPU computational time for (a) CTPS C_a^2 and (b) CP C^2 (Case 2).

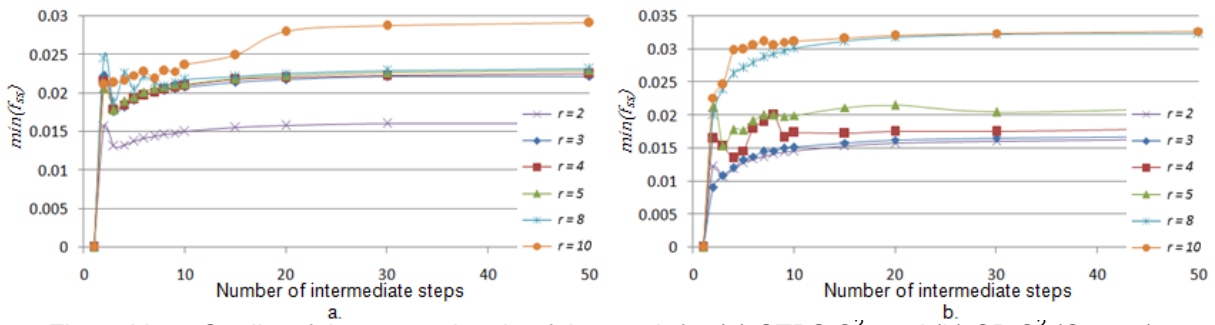


Figure V.17. Quality of the worst triangle of the mesh for (a) CTPS C_a^2 and (b) CP C^2 (Case 3).

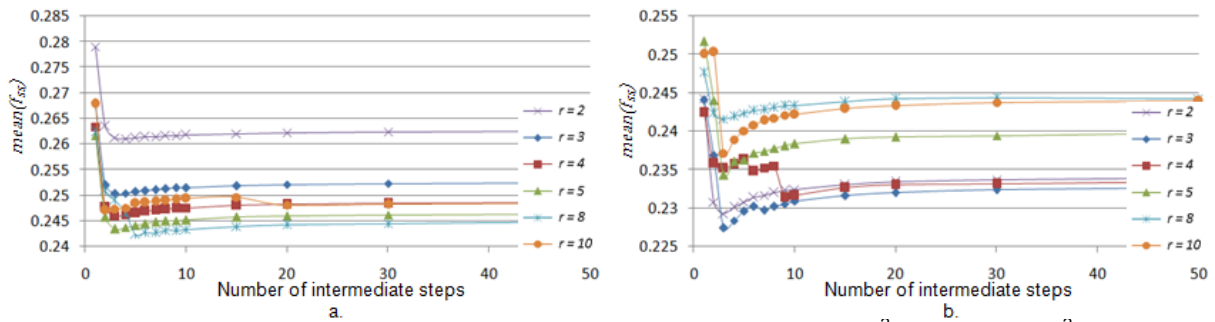


Figure V.18. The mean quality of all triangles in the mesh for (a) CTPS C_a^2 and (b) CP C^2 (Case 3).

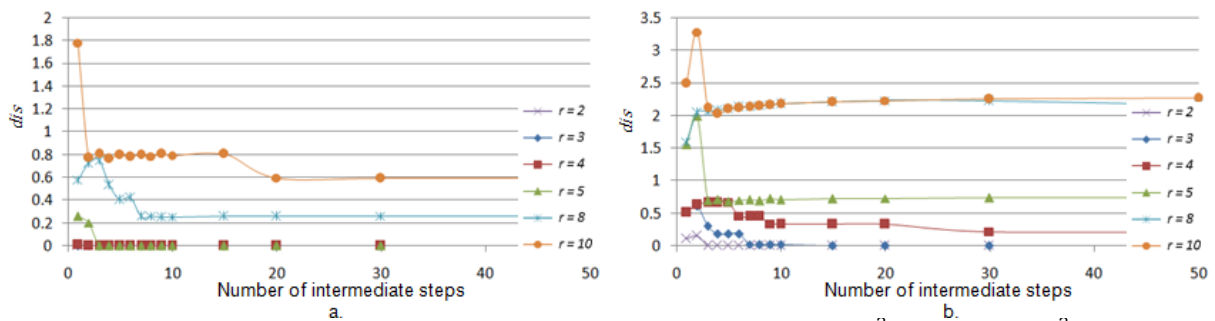


Figure V.19. Accuracy displacement of the control points for (a) CTPS C_a^2 and (b) CP C^2 (Case 3).

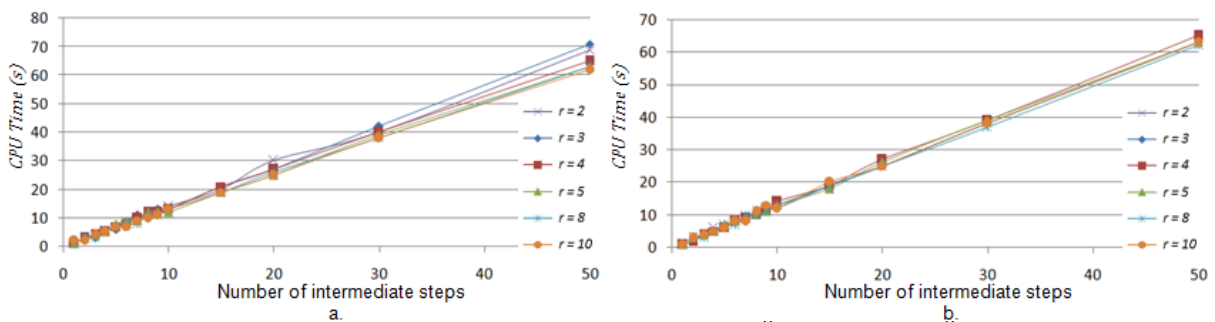


Figure V.20. CPU computational time for (a) CTPS C_a^2 and (b) CP C^2 (Case 3).

The analysis of the experimental results presented in Figure V.9 to Figure V.20 leads to the following conclusions:

1. For all scenarios and for both RBFs (CTPS C_a^2 and CP C^2), the parameter $min(f_{ss})$ reaches its maximum value when the support radius r is increased starting from values of $r = 8$). However, excessively increasing the support radius r (up to 10) also leads to a greater error in terms of approximation precision (parameter dis). In the context of mesh warping, such a behavior would translate into an imprecise feature vertex alignment. Thus, a

trade-off between mesh quality and displacement accuracy has to be determined. In our examples, values of r inferior to 5 seem to provide a fair compromise.

2. When more intermediary steps are used to deform the mesh, we notice that for both RBFs the minimum value of f_{ss} is increasing. This shows the interest of considering a step-by-step approach. However, when analyzing the average values of f_{ss} , we can observe that the mean quality of the mesh is reducing when increasing number of steps. This can be explained by the fact that when the rectangle is moved directly (a single step), a large part of the original mesh remains quasi-constant, and only a small part of triangles is distorted. On the contrary, when the rectangle is displaced through more steps a greater area of the original mesh is smoothly deformed since, at each step, more triangles are captured within the radius of influence of the considered RBF. This phenomenon is illustrated in Figure V.21. However, a larger number of steps is preferable since when performing single step deformations the affected triangles are highly distorted (*cf.* values of parameter $\min(f_{ss})$).

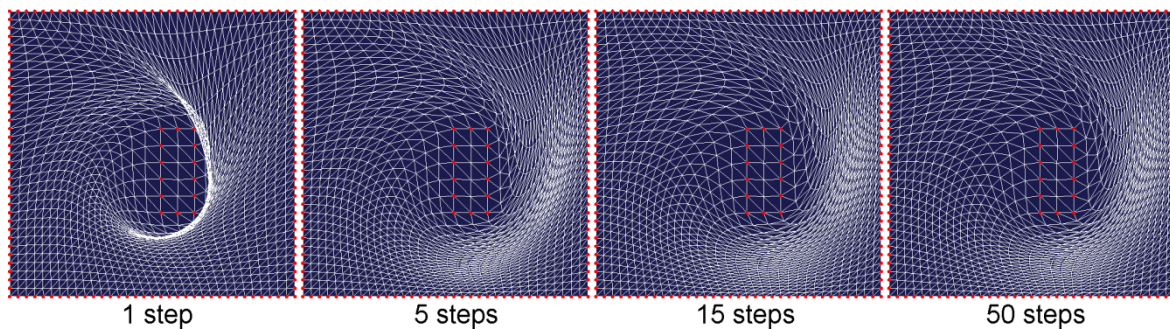


Figure V.21. The influence of the number of steps on the deformed mesh.

3. In the case of a strong deformation such as the one considered in case 3, the impact of the intermediary number of steps used for warping it is even more important because when this parameter has a low value (one or two steps) the mesh triangles overlap.

4. If we highly increase the number of intermediate steps, we will not obtain any considerable improvements on the overall quality. Thus, starting from a number of 10-15 steps, the results are quite equivalent.

5. The CPU computational time is, without surprise and in all cases, linearly affected by the number of deformation steps. Concerning the support radius r , it has a negligible impact on the computation times. The computation times reported here have been obtained on an Intel Core2Duo machine at 2,13GHz and with 3GB Ram under a Windows XP SP2 platform.

Based on these considerations, we have selected for further evaluations, a value of 4 for the support radius r . The chosen value establishes a trade-off between the mesh deformation quality expressed through the metric f_{ss} and the accuracy displacement given by the dis parameter. Concerning the number of intermediate steps, it was set to 15. Figure V.22

presents an example of the deformation evolution when applying the CTPS C_a^2 function with the support radius r set to 4 and the number of intermediary steps equal to 15.

Let us now compare both RBF and Laplacian deformation methods. Table V.2 summarizes the $\min(f_{ss})$ quality metric, $\text{mean}(f_{ss})$, displacement accuracy (dis), and computational time for the considered warping methods in all three scenarios presented earlier. Some visual results are presented in Figure V.23.

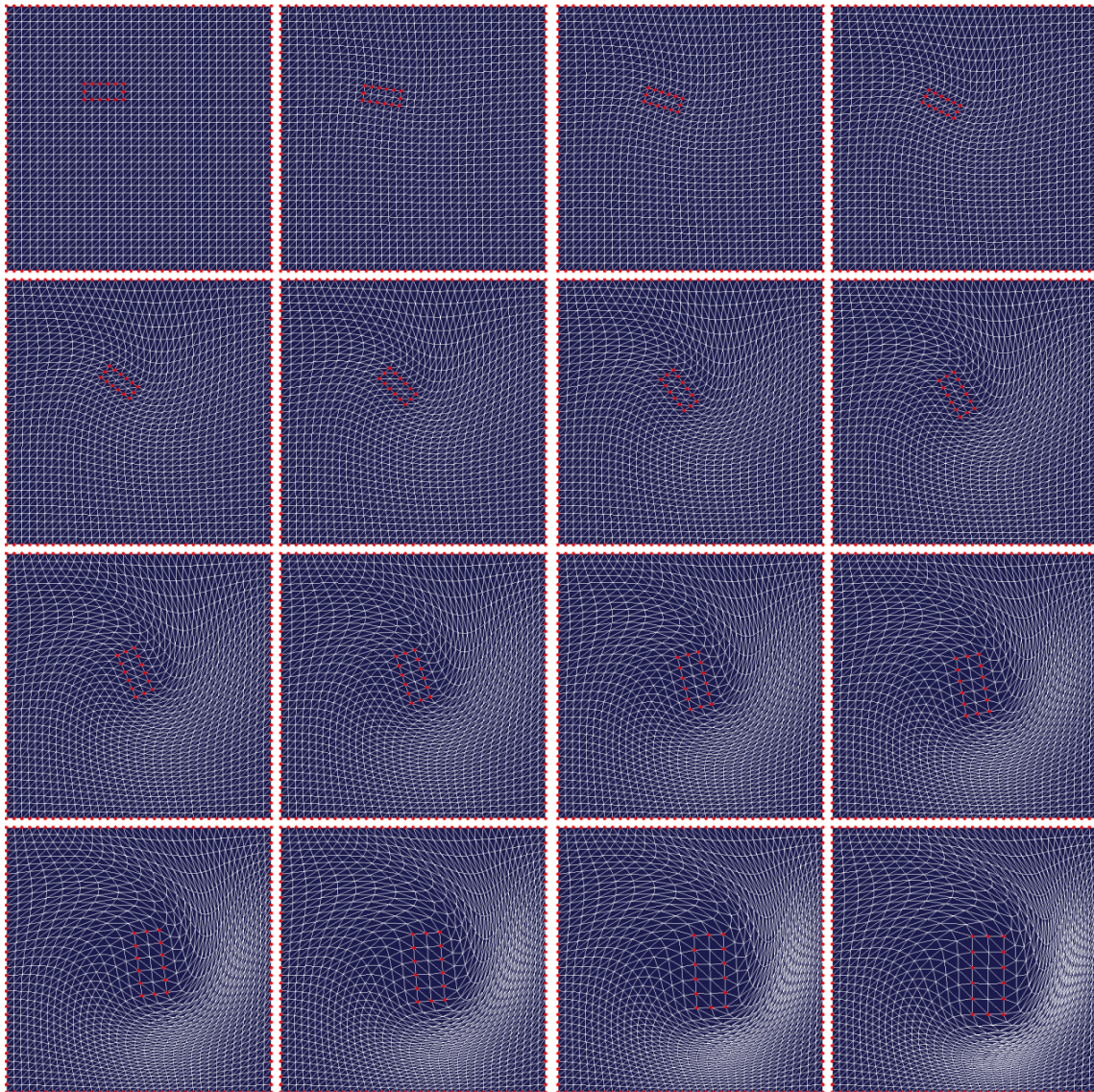


Figure V.22. RBF mesh deformation on steps.

After evaluating the experimental results, a first conclusion can be derived: when using the UFLC and TLC techniques in the planar domain a mesh overlapping is produced. This is caused by the strict conditions imposed on the control points (*i.e.* they should reach exactly the final destination). The classical Laplacian method leads to a valid deformed mesh, but with relatively high distortions (50% greater than the RBF-related distortions) and also large values of the accuracy parameter dis .

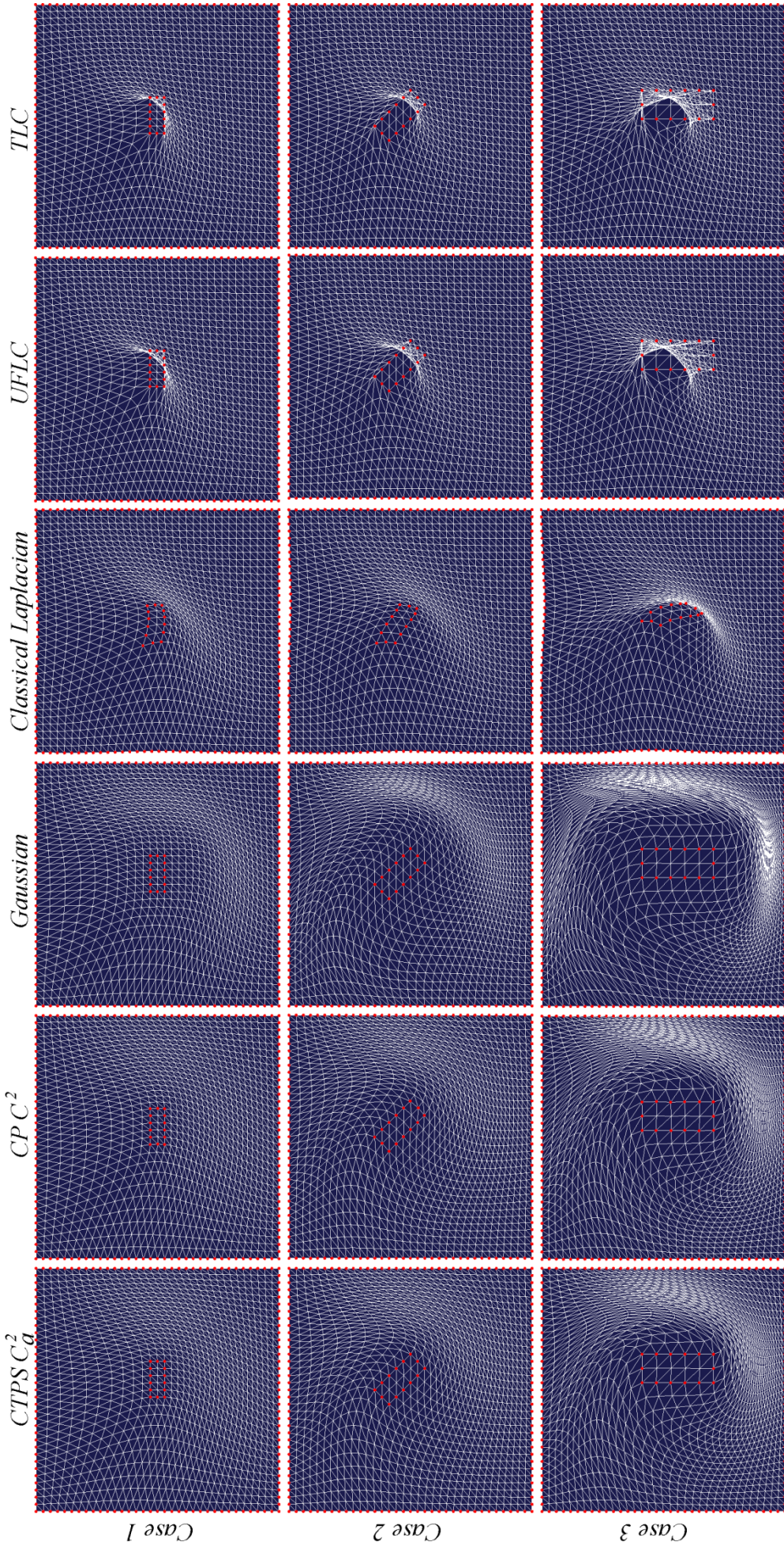


Figure V.23. Visual analysis of mesh deformation in 2D space.

Table V.2. Mesh deformation quality analysis for 2D test cases.

Method	$\min(f_{ss})$	$\text{mean}(f_{ss})$	dis	Inverted triangles	CPU Time (sec)
Test case 1					
RBF - CTPS C_a^2	0.148	0.271	$6.51 \cdot 10^{-5}$	0	18.9
RBF - CP C^2	0.152	0.270	0.230	0	21.6
RBF - Gaussian	0.103	0.258	0.013	0	20.7
Classical Laplacian	0.091	0.366	6.030	0	0.9
UFLC	0	0.379	$9.53 \cdot 10^{-9}$	21	1
TLC	0	0.381	$1.41 \cdot 10^{-9}$	15	1.3
Test case 2					
RBF - CTPS C_a^2	0.091	0.271	$1.13 \cdot 10^{-5}$	0	19.1
RBF - CP C^2	0.089	0.271	0.456	0	17.8
RBF - Gaussian	0.051	0.304	0.031	0	20.8
Classical Laplacian	0.106	0.373	14.276	0	0.9
UFLC	0	0.386	$1.01 \cdot 10^{-8}$	23	1
TLC	0	0.388	$1.56 \cdot 10^{-9}$	17	1.9
Test case 3					
RBF - CTPS C_a^2	0.021	0.247	$8.34 \cdot 10^{-6}$	0	21.1
RBF - CP C^2	0.017	0.232	0.334	0	18.6
RBF - Gaussian	0	0.198	0.043	22	21.3
Classical Laplacian	0.001	0.331	20.501	0	0.8
UFLC	0	0.354	$6.26 \cdot 10^{-9}$	60	0.9
TLC	0	0.355	$1.03 \cdot 10^{-9}$	51	1.5

Concerning the radial basis functions, CTPS C2a and CP C2 return comparable results in terms of $\min(f_{ss})$ and $\text{mean}(f_{ss})$, but a higher accuracy is obtained for CTPS C2a. In this case, the final position of the control points is reached with an error inferior to 10^{-4} . When warping the mesh using the RBF Gaussian function the experimental results show the lowest values for the $\min(f_{ss})$ compared to any other analysed RBF function. The results are even more disturbing in the third scenario (high amplitude deformation) when the method is not able to conduct to a valid deformation (triangles fold-over).

When comparing the computational times, it can be observed that the Laplacian functions have the lowest processing requirements and in most of the cases the final result is obtained in less than a second. This observation can be motivated by the total number of intermediary steps involved in the deformation: for the Laplacian algorithm the displacement is made directly while for the RBF functions we used 15 steps in order to achieve a high quality for the deformation. As it can be noticed from Table V.2, all RBF functions are computed in almost the same CPU time (about 20 seconds)

V.4.2. Deformation in 3D test cases

The second set of test scenarios consist of a square domain with an inner rectangle, characterized by the same parameters as presented in Section V.4.1. that undergoes different geometric transforms, this time in the 3D space, out of the plane where the mesh grid is defined. Here again, we consider the following three different cases:

- a translation in space (along the z axis): The control rectangle is translated in the x and y directions by 6 units each and in the z direction by 10 units (a unit corresponds to a sampling interval of the considered mesh grid);
- a translation combined with a moderate rotation and scaling: here, the control rectangle is translated in the x direction by 7 units, 2 units in the y direction and 10 units in the z direction. Furthermore, a rotation of 45° clockwise and a scaling by a factor $\sqrt{2}$ are applied;
- a high amplitude rotation, translation and scaling: in this case, the control rectangle is translated in the x direction by 12 units, 4 units in the y direction and 10 units in the z direction. Furthermore, a rotation of 90° clockwise and a scaling of a factor 2 are applied.

We started our experiments by visually analyzing the impact of the support radius r and the number of intermediary steps over the mesh deformation. Figure V.24 presents some results obtained for the CTPS C_a^2 function for the third test scenario. We can observe that smoother deformation fields are obtained when increasing both the support radius and the number of intermediate steps. In the same time, the spatial extent of the deformation field is also increased in such cases.

Figure V.25, Figure V.26 and Table V.3 present the comparative evaluation for the considered methods, in all scenarios proposed for the 3D space.

In terms of $\min(f_{ss})$ and $\text{mean}(f_{ss})$ quality metrics the best results are obtained by the classical Laplacian method. For example, in the first scenario, the classical Laplacian returns a value for $\min(f_{ss})$ of 0.29, which is with 13% higher than the best result acquired by a RBF function (CP C^2). Concerning the $\text{mean}(f_{ss})$ parameter, the Laplacian improves the results with more than 10% compared with the Gaussian function which returns now the best performances from all RBF techniques. However, as in the 2D case, the displacement accuracy of this method is poor (with a parameter dis of one order of magnitude greater than in the case of RBFs).

If in the 2D case, the mesh deformation techniques based on UFLC and TLC led to triangle overlapping, in the 3D case this problem is avoided since the warping is performed in space. In terms of (f_{ss}) , these methods return comparable results as the ones obtained by the classical Laplacian, but with a $\min(f_{ss})$ inferior with more than 68%.

These results can be explained by the fact that the overall structure of the mesh remains almost unmodified, only the regions near the displaced rectangle are significantly altered. As

it can be noticed UFLC and TLC methods have the benefit of the most accurate mesh displacement (*i.e.*, the rectangle is displaced exactly into the desired position).

Table V.3. Mesh deformation quality analysis for 3D test cases.

Method	$\min(f_{ss})$	$\text{mean}(f_{ss})$	dis	CPU Time (sec)
Test case 1				
RBF – CTPS C_a^2	0.232	0.598	0.001	19.8
RBF – CP C^2	0.251	0.597	0.249	19.3
RBF – Gaussian	0.135	0.605	0.010	23.1
Classical Laplacian	0.291	0.672	6.031	0.21
UFLC	0.075	0.676	$5.50 \cdot 10^{-9}$	0.19
TLC	0.091	0.672	$1.01 \cdot 10^{-9}$	0.49
Test case 2				
RBF – CTPS C_a^2	0.228	0.583	$5.64 \cdot 10^{-6}$	21.2
RBF – CP C^2	0.227	0.577	0.331	18.4
RBF – Gaussian	0.127	0.544	0.020	21.1
Classical Laplacian	0.301	0.675	9.488	0.22
UFLC	0.081	0.683	$5.58 \cdot 10^{-9}$	0.21
TLC	0.092	0.679	$1.03 \cdot 10^{-9}$	0.73
Test case 3				
RBF – CTPS C_a^2	0.118	0.465	$3.36 \cdot 10^{-6}$	19.7
RBF – CP C^2	0.104	0.438	0.168	19.1
RBF – Gaussian	0.064	0.426	0.012	23.8
Classical Laplacian	0.228	0.613	14.751	1.12
UFLC	0.047	0.623	$4.53 \cdot 10^{-9}$	0.31
TLC	0.045	0.618	$8.82 \cdot 10^{-10}$	0.89

If we analyze only the radial basis functions we observe that the best results, in terms of $\min(f_{ss})$ and $\text{mean}(f_{ss})$, are obtained for the CTPS C_a^2 function which offers also the best displacement accuracy.

The experimental results show that RBF method (and, in particular the CTPS C_a^2 function) offer a good compromise between displacement accuracy and mesh quality. We have thus retained this method for performing warping of the parametric domain meshes.

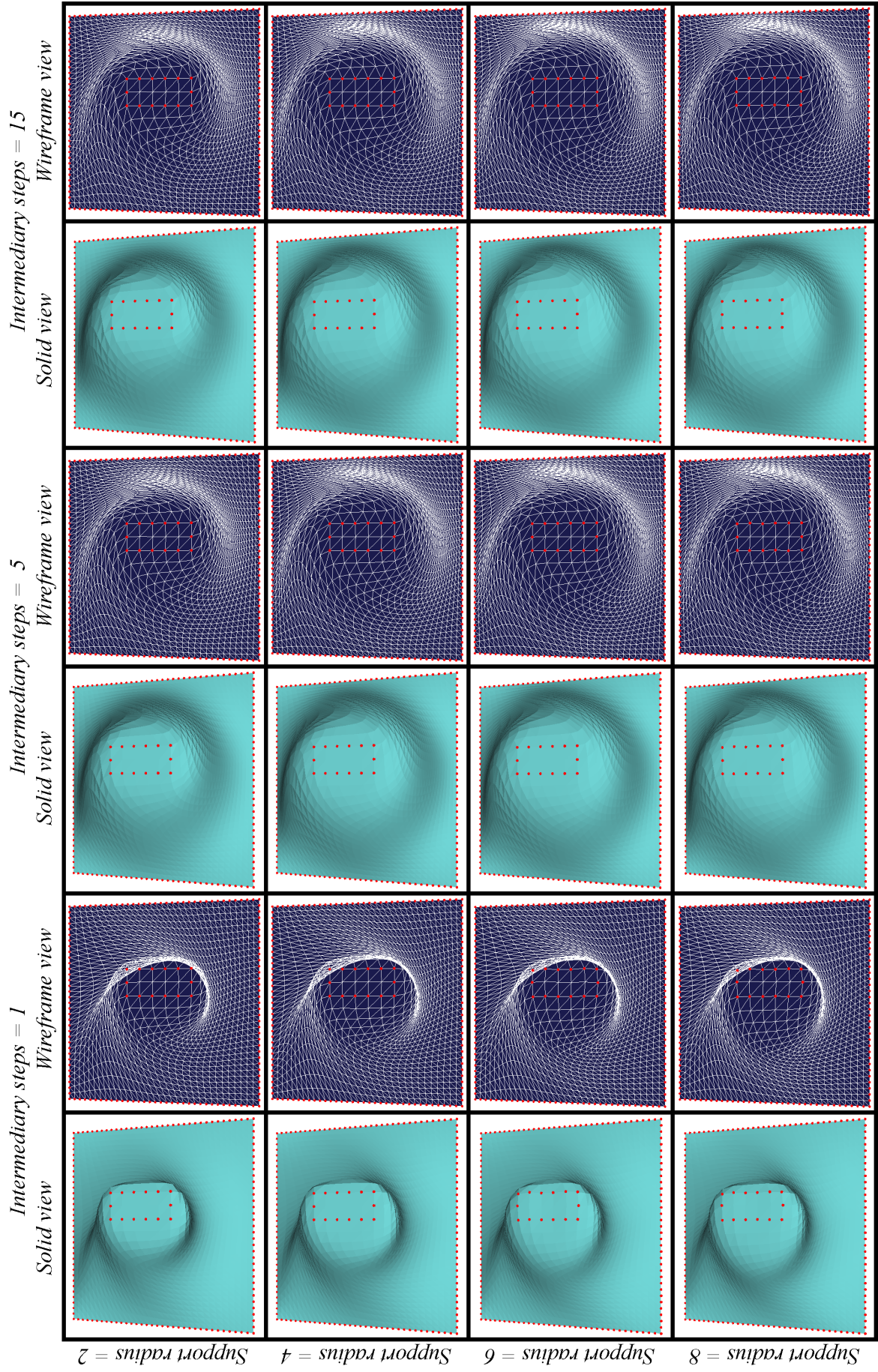


Figure V.24. The impact of the support radius r and the number of intermediary steps over the mesh deformation for CTPS C^2_a function (Test case 3).

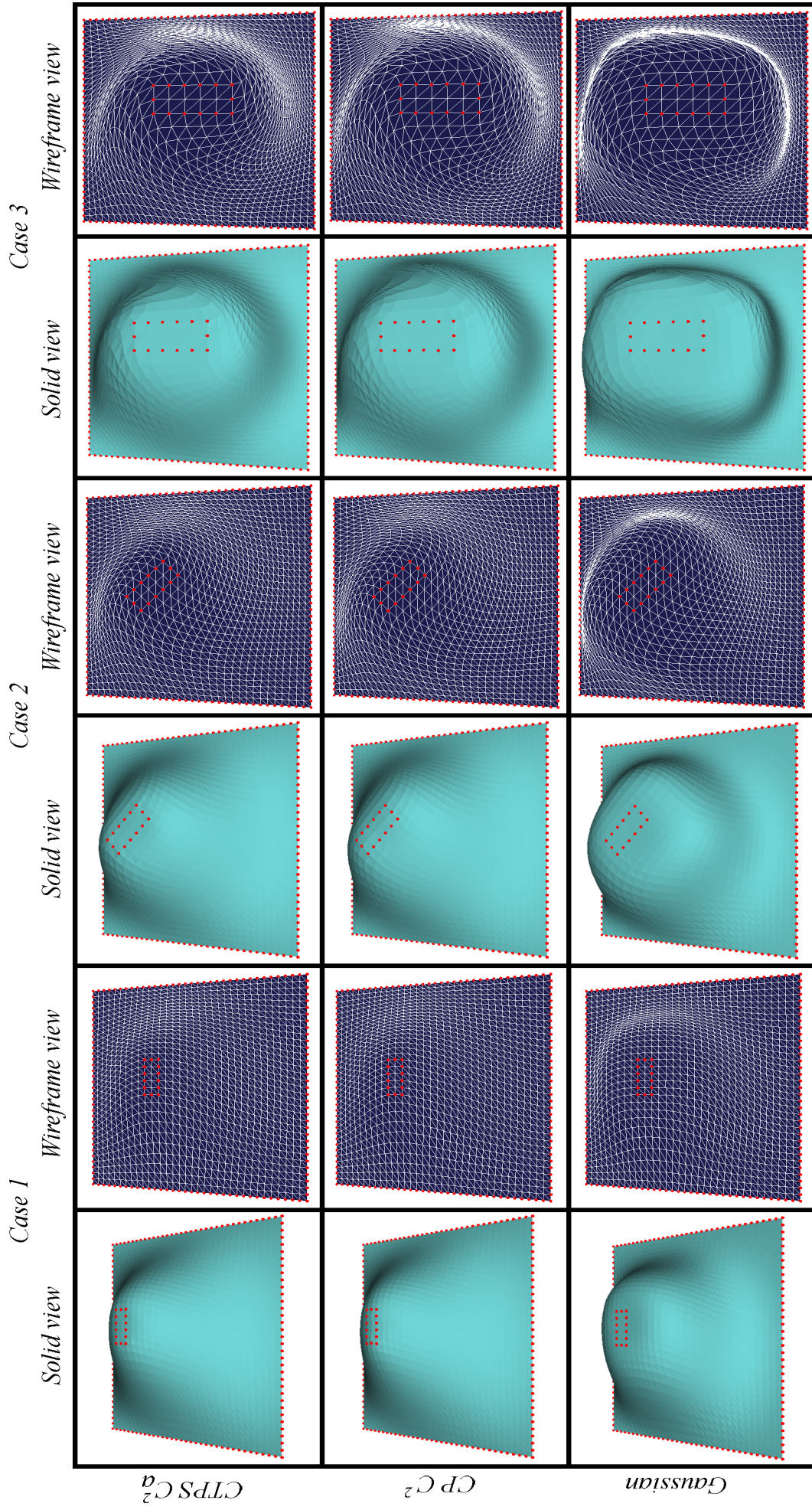


Figure V.25. Visual analysis of mesh deformation in 3D case for the RBF functions: CTPS C^2_a , CP C^2 and Gaussian.

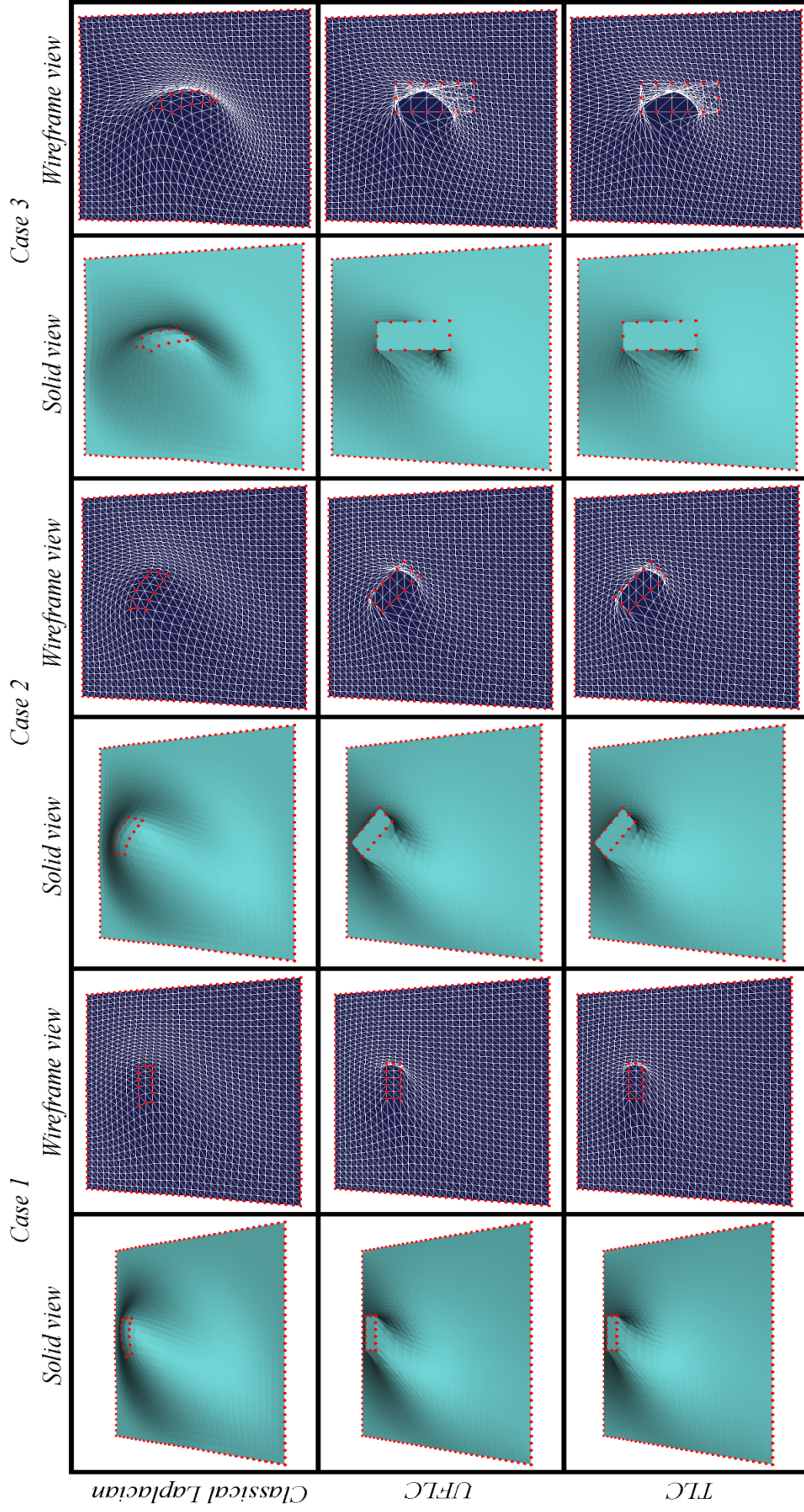


Figure V.26. Visual analysis of mesh deformation in 3D case for Classical Laplacian coordinates, UFLC and TLC methods.

V.5. FEATURE ALIGNMENT BASED ON MESH WARPING

We have chosen to apply CTPS C_a^2 function in steps because a direct RBF implementation would lead to a deformed mesh with the feature vertices placed at the right location, but with the mesh surface not on the unit sphere.

The algorithm splits the distance between any feature-pair $(p_i^{S'}, p_i^{T'})$ proportionally with the maximum geodesic distance between any pair and apply for each interval the RBF algorithm that deform the meshes accordingly. Due to the movement, not all vertices may be on the sphere anymore. We guarantee that the final embedding remains valid we propose projecting the mesh back on the unit sphere after each intermediary step.

Additionally, in order to further reduce the distortion rate we constrain that both source and target feature vertices to move in their middle position $(p_i^{S'} + p_i^{T'})/2$.

Figure V.27 and Figure V.28 present two examples of spherical warping. Each example contains two 3D closed models with feature vertices already specified by user, their initial spherical mappings and their final embeddings (where pair vertices are put in correspondence on the parametric domain).

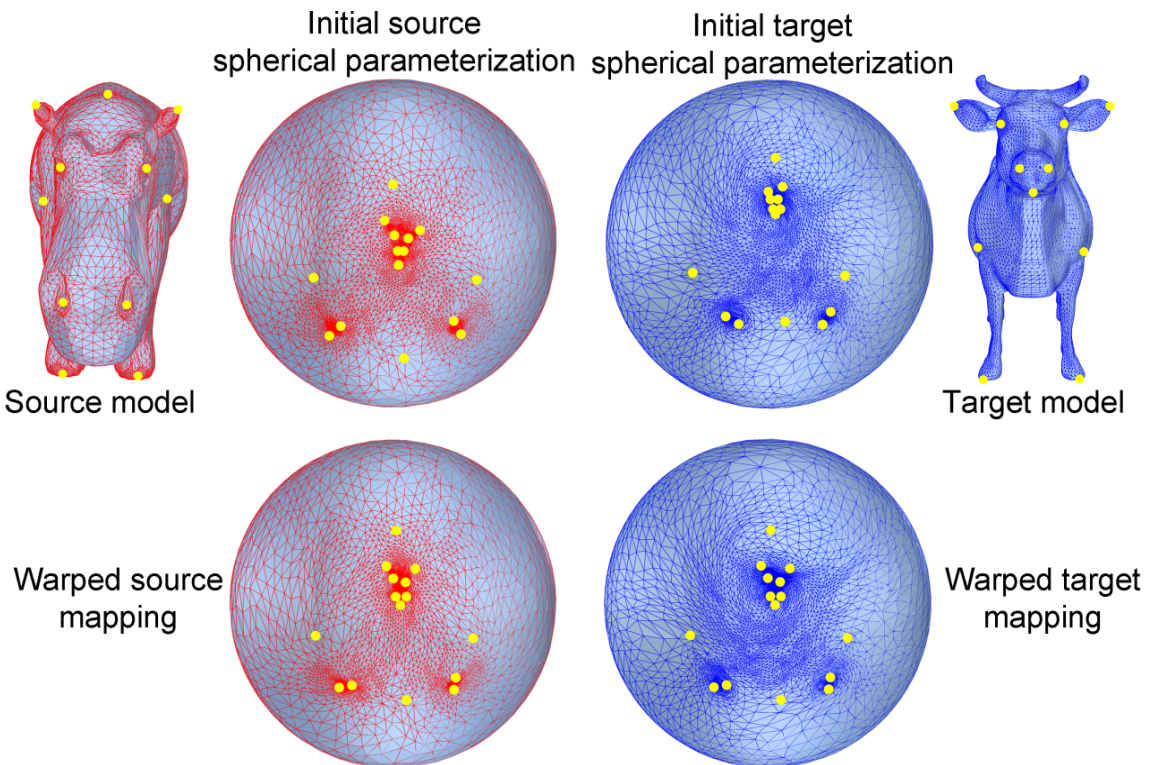


Figure V.27. Feature vertices correspondence through spherical embeddings warping (Hipo-Cow case).

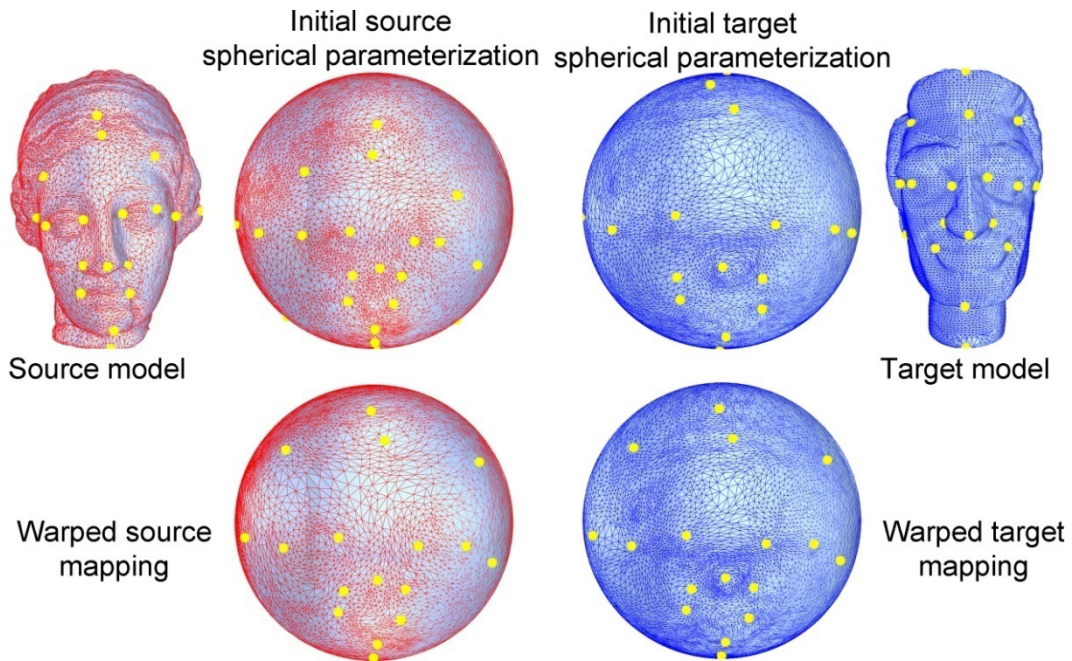


Figure V.28. Feature vertices correspondence through spherical embeddings warping (Igea-ManHead case).

V.6. CONCLUSIONS

In this chapter, we have first provided an overview of the various methods of mesh deformation. After the basic concepts were outlined we have selected for a more detailed evaluation the radial basis function method and the Laplacian coordinates technique, which are the most well-suited for mesh warping purposes.

In order to evaluate the capacity of RBF and Laplacian coordinates as mesh movement strategies we employed several test cases in both 2D and 3D space and we have analyzed the algorithms behavior in terms of deformation quality and displacement accuracy.

Regarding the radial basis functions we have demonstrated that a deformation through steps returns better results than a direct one. The higher the number of intermediary steps the higher quality is obtained, but with the cost of more computational resources. However, after a number of 10 to 15 steps, the overall quality will not increase considerably.

From the six methods considered for evaluation, the Uniform Fix Laplacian coordinate (UFLC) and Tangential Laplacian coordinate (TLC) deformation methods offers the highest values for the $mean(f_{ss})$ and dis metrics when warping in the 3D space. However, the distortions near the control points are unacceptable compared with other methods and, moreover, in the 2D deformation scenarios, such techniques lead to fold-overs.

As a consequence, we have retained for mesh warping purposes, the CTPS C_a^2 RBF method, which offers a good compromise between displacement accuracy and overall quality of the deformed mesh.

VI. SUPERMESH CONSTRUCTION AND INTERPOLATION

Summary: *This chapter first provides an overview of the algorithms used in the state of the art for the creation of the so-called supermesh structure. We introduce a novel method that avoids the classical edge-to-edge intersection procedure. The supermesh is constructed with the help of progressive subdivisions, accordingly to the topology of both source and target input models. A short overview of the mesh interpolation techniques is also supplied. Finally, we present our graphical user interface elaborated for mesh morphing purposes in this thesis.*

VI.1. INTRODUCTION

Once the two input models are parameterized onto a common domain and the main features of the objects are properly aligned, the next step required in a morphing framework is to establish a one-to-one correspondence between the shapes to be morphed. In order to accomplish this task, the two embeddings need to be overlapped. In addition, a new mesh structure that can represent the connectivities of both models need to be constructed. The resulting mesh is called supermesh or metamesh (SM). The main principle behind the supermesh construction is to merge the two topologies into a single one by inserting edges of the target model into the source structure.

The main advantage of the metamesh consists of its property of sharing both the source and target topologies. Thus, the two model shapes can be accurately approximated by the new mesh structure. The metamesh will represent in a morphing sequence the source model at the first frame and the target model at the last frame. Thus, for each vertex p_i of the supermesh we must determine two positions: a first one relative to the source shape (p_i^0) and a second one corresponding to the target shape (p_i^1). For intermediary frames, the vertices positions of each supermesh vertex are interpolated between the initial and final states. However, determining appropriate trajectories for connecting the initial position p_i^0 to the final position p_i^1 still remains a challenging issue. The process of transforming the shape of the source mesh into the shape of the target mesh (and vice versa), based on the established vertex trajectories, is called mesh interpolation.

The following sections describe the various approaches of supermesh construction proposed in the literature as well as the different mesh interpolation methods proposed.

VI.2. TOPOLOGY MERGING FOR MESH MORPHING

The concept of topology merging for mesh morphing was first introduced in [Ken92]. Let us already note that the source and target connectivities cannot be directly merged in the original space, since the edges are in the general case nonparallel and nonintersecting. Instead, the process can be performed within the parametric domain, where the edges lie on a planar or spherical surface. Therefore, the metamesh construction can be achieved by: (1) overlapping the parameterizations; (2) performing exhaustively the edge intersection of the two meshes. The last phase requires a local triangulation that allows obtaining the shared triangular topology, as illustrated in Figure VI.1 for the planar parameterization case.

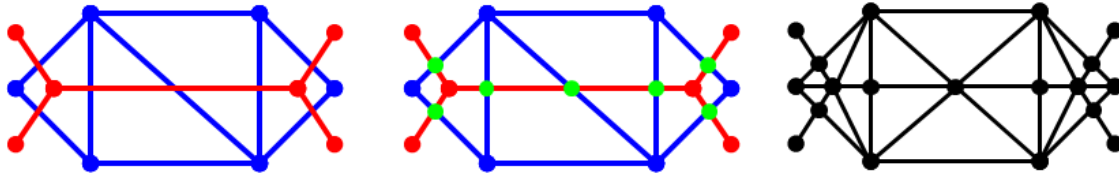


Figure VI.1. Constructing the supermesh: (a) embedding of the two connectivities in the common parametric domain; (b) Edge to edge intersection; (c) Triangulation.

The merging algorithm proposed by Kent *et al.* [Ken92] is based on the assumption that, after the overlapping the parameterizations, no parametric vertices of the two models are coincident, and no parametric vertex of one model lies on an edge on the other model. In order to illustrate the proposed procedure, let us consider the example presented in Figure VI.2. Here, the blue color represents source elements while the red color the target ones.

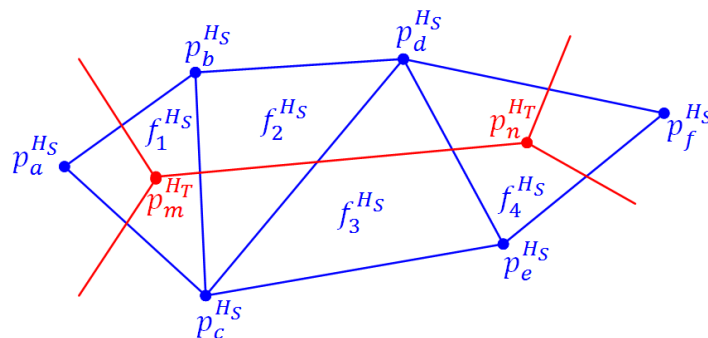


Figure VI.2. Edge intersection algorithm of [Ken92].

Starting with an arbitrary edge $e_{m,n}^{H_T}$ from the target mapping (H_T), with the endpoints $p_m^{H_T}$ and $p_n^{H_T}$, the triangle of the source parameterization (H_S) in which the vertex $p_m^{H_T}$ lies is first determined, with the help of a *point in triangle* test. In our example, the face $f_1^{H_S}$ is thus determined. The incident target edges to $p_m^{H_T}$ are then added to a list of edges to be processed, called the work list. The edge $e_{m,n}^{H_T}$ is the first one in this list. Since it is known that $p_m^{H_T}$ lies inside the triangle $f_1^{H_S}$, the first intersection of the edge $e_{m,n}^{H_T}$ should be with an edge of this face. Thus, $e_{a,b}^{H_S}, e_{b,c}^{H_S}$ and $e_{c,a}^{H_S}$ are added to a list of candidate edges that $e_{m,n}^{H_T}$ might intersect. In the case presented in Figure VI.2, $e_{m,n}^{H_T}$ intersects $e_{b,c}^{H_S}$. Using the topology of H_S it can be determined that $e_{m,n}^{H_T}$ crosses the triangle $f_2^{H_S}$. Thus, $e_{b,d}^{H_S}$ and $e_{c,d}^{H_S}$ are considered potential edges that $e_{m,n}^{H_T}$ might intersect. Similarly, at the intersection of $e_{c,d}^{H_S}$ with $e_{m,n}^{H_T}$, edge $e_{m,n}^{H_T}$ crosses the triangle $f_3^{H_S}$ and edges $e_{d,e}^{H_S}$ and $e_{c,e}^{H_S}$ are added to the candidate list. At the intersection of $e_{d,e}^{H_S}$ and $e_{m,n}^{H_T}$, edge $e_{m,n}^{H_T}$ crosses the face $f_4^{H_S}$. Since $e_{m,n}^{H_T}$ does not intersect either $e_{d,f}^{H_S}$ or $e_{e,f}^{H_S}$, vertex $p_n^{H_T}$ must lie on face $f_4^{H_S}$. This fact is recorded and the untreated target edges incident to $p_n^{H_T}$ are added to the work list. The above technique is repeated for each edge in the work list until it remains empty. At the end of the process, a new mesh structure is obtained. However, the resulting structure will contain faces with more

than three edges. In order to obtain a valid mesh structure, a heuristic mesh triangulation procedure is finally applied.

Next, the algorithm establishes, using the point-in-triangle test, which triangles of H_T contain each vertex of H_S . These information together with those that indicates which face of H_S contains each vertex of H_T are used to determine where the vertices of one model map onto the surface of the other. This task is accomplished using the barycentric coordinates. If we consider for example the vertex $p_m^{H_T}$ as presented in Figure VI.2, once the barycentric coordinates α , β and γ are computed relatively to face $f_1^{H_S}$, we can establish the position of p_m^S on the original source model shape (M_S) as:

$$p_m^S = \alpha p_a^S + \beta p_b^S + \gamma p_c^S \quad (\text{VI.1})$$

where p_a^S , p_b^S and p_c^S are the vertex positions on the original source surface.

The drawback of the proposed method is related to the underlying non-coincidence hypothesis of source and target mesh vertices, which limits its applicability in practice.

In order to overcome such a limitation, Kanai *et al.* [Kan98] propose a slightly different method which is able to take into account coincident vertices/edges. In order to avoid numerical errors, the coincident vertices are first determined. The source and target parameterizations H_S and H_T are then re-calculated by maintaining these vertices fixed to an average position. The operation is iterated until no coincident vertices are generated. The case of a vertex of one embedding lying on an edge of the other embedding is treated in a similar manner.

After this pre-processing step, the supermesh construction is performed in a similar manner with the one proposed in [Ken92]. In addition, in order to speed up the searching of a face including a vertex, the authors use a spatial partitioning procedure based on a quad-tree data structure [Fol90].

The mesh retriangulation is realized as follows: The adjacent edges to a vertex are sorted in a counterclockwise order. If two consecutive edges $e_{i,j}$ and $e_{j,k}$ are not already connected by another edge $e_{i,k}$, then the edge $e_{i,k}$ is created as well as a new face $f(i,j,k)$. This operation is performed until all edges have a triangular face on both sides.

In order to solve the coincidence problem, Alexa [Ale00] proposes to use a *symbolic perturbation scheme* as the one described in [Her90] which makes it possible to avoid the cases when a vertex of one embedding lies on a vertex/edge on the other graph.

Since the method proposed by Alexa[Ale00] parameterizes the models onto the unit sphere, the problem of edge-to-edge intersection transforms into an arc-to-arc intersection. Here, the

edge between two points p_1^H and p_2^H on the sphere should be seen as the shorter arc of the circle with radius 1 and the same center as the sphere, passing through the points. The intersection point p_I^H between two edges $e(p_1^H, p_2^H)$ and $e(p_3^H, p_4^H)$ is established using the following equation:

$$p_I^H = \pm(p_1^H \times p_2^H) \times (p_3^H \times p_4^H) \quad (\text{VI.2})$$

Actually, p_I^H specifies the two positions where the great circles defined by the two considered edges intersect. The following system of equation has to be solved in order to determine whether the intersections lie on both arcs:

$$\begin{aligned} t_a p_I^H &= p_1^H + s_a (p_2^H - p_1^H) \\ t_b p_I^H &= p_3^H + s_b (p_4^H - p_3^H) \end{aligned} \quad (\text{VI.3})$$

where t_a , t_b and s_a , s_b are unknowns that specify if the intersection is a common point of the two arcs. If $s_a, s_b \in (0, 1)$ and $t_a, t_b > 0$, then the two arcs intersect in p_I^H .

The rest of the supermesh construction process remains similar with the previous approaches, only with the difference that Alexa uses a more sophisticated data structure to represent the models, which consists of a double connected edge list.

In [Urt04], authors re-visit the reference concept of metamesh construction introduced by Kent *et al.* [Ken92], and propose to improve it by taking into consideration the known vertex positions of the 3D shapes. If the metamesh construction process starts with the source topology, the target vertices are added progressively into the structure as well as the new points resulted after the edge intersection. When the supermesh takes the shape of the source model, these vertices are placed on the source faces forcing them to be flat, while their real position should be at some distance above or below the considered face.

If we consider for example the vertex p_m^{Hr} as presented in Figure VI.2, once the barycentric coordinates α , β and γ are computed relatively to face f_1^{Hs} , in contrast with equation (VI.1), the 3D coordinates of p_m relatively to the original source mesh shape are computed as:

$$p_m^S = \alpha(p_a^S + n\rho_1) + \beta(p_b^S + n\rho_2) + \gamma(p_c^S + n\rho_3) \quad (\text{VI.4})$$

where the vertex p_m lies on the source surface on the face $f_1^S(p_a^S, p_b^S, p_c^S)$, n is the vertex normal of p_m and ρ_i ($i = 1, 2, 3$) is given by:

$$\rho_i = d_i \cos\phi_i + d_i \sqrt{\frac{(1 - \cos^2\phi_i)(1 - \cos\phi_i)}{1 + \cos\phi_i}} \quad (\text{VI.5})$$

$$\cos\phi_i \cong n \cdot n_i \quad ; \quad \cos\phi_i \cong \frac{n \cdot d_i}{\|d_i\|} \quad (\text{VI.6})$$

where n_i is the vertex normal of p_i^S (i corresponding here to indices a, b or c), while d_i is the vector between one vertex of $f_1^S(p_a^S, p_b^S, p_c^S)$ and the projection of the new vertex on the face.

In [Lee03], authors introduce a new approach called SMCC (*Structures of Minimal Contour Coverage*) that handles all coincident, degenerate cases with the help of a simple data structure. The merging algorithm overlays each target edge on the source topology. Depending on the place where the edge endpoints and the new vertices lie on the metamesh triangles different cases of intersection are distinguished. When an edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ is overlaid on H_S , this edge can be split into several line segments by the triangles f^{H_S} .

This principle is illustrated in Figure VI.3, where the green dots represent vertices obtained after the intersection. The following three cases can be encountered for the point $p_{start}^{H_T}$: (1) $p_{start}^{H_T}$ lies inside a triangle f^{H_S} , (2) $p_{start}^{H_T}$ coincides with another vertex p^{H_S} , and (3) $p_{start}^{H_T}$ lies on an edge e^{H_S} . If $p_{end}^{H_T}$ is not in the same triangle as $p_{start}^{H_T}$, then the edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ is split and the new intersection point becomes a new $p_{start}^{H_T}$. This process is repeated until $p_{end}^{H_T}$ will find on the same triangle as $p_{start}^{H_T}$ (on a vertex, on an edge or inside the same triangle).

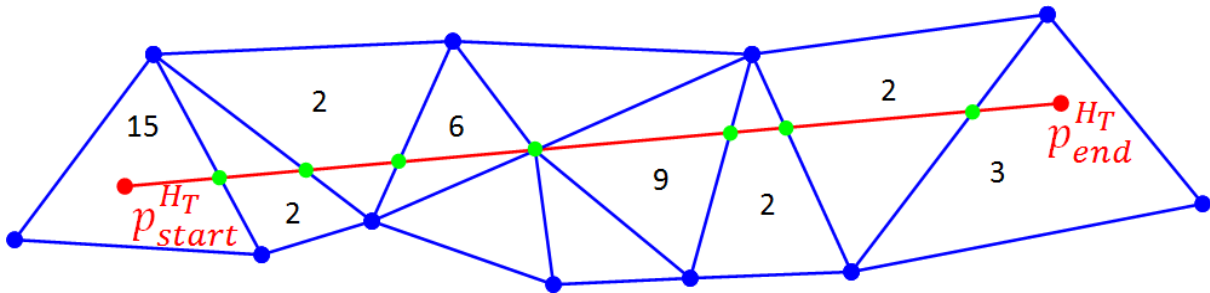


Figure VI.3. Edge intersection scheme labeled according to the SMCC algorithm [Lee03].

Based on this principle, 18 different cases of intersections can be identified. They are illustrated in Figure VI.4:

- Case 1: $p_{start}^{H_T}$ lies on an edge of triangle f^{H_S} , $p_{end}^{H_T}$ is outside of f^{H_S} and at least one intersection point p_I is obtained.
- Case 2: $p_{start}^{H_T}$ lies on an edge of triangle f^{H_S} , $p_{end}^{H_T}$ belongs to another edge of f^{H_S} and no additional intersection points exist.
- Case 3: $p_{start}^{H_T}$ lies on an edge of triangle f^{H_S} , $p_{end}^{H_T}$ is inside of f^{H_S} and no additional intersection points exist.
- Case 4: $p_{start}^{H_T}$ lies on an edge of triangle f^{H_S} , $p_{end}^{H_T}$ is outside of f^{H_S} and the inserted edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ lies on the edge where $p_{start}^{H_T}$ is situated.
- Case 5: $p_{start}^{H_T}$ lies on an edge of triangle f^{H_S} , $p_{end}^{H_T}$ coincides with one of the f^{H_S} vertices, and the inserted edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ lies on the edge where $p_{start}^{H_T}$ is situated.
- Case 6: $p_{start}^{H_T}$ lies on an edge of triangle f^{H_S} , $p_{end}^{H_T}$ coincides with one of the f^{H_S} vertices, but the inserted edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ does not lie on the edge where $p_{start}^{H_T}$ is situated.

- Case 7: $p_{start}^{H_T}$ and $p_{end}^{H_T}$ lies on the same edge of triangle f^{H_S} .
- Case 8: $p_{start}^{H_T}$ coincides with one of the f^{H_S} vertices, $p_{end}^{H_T}$ is outside of f^{H_S} and at least one intersection point p_I is obtained.
- Case 9: $p_{start}^{H_T}$ coincides with one of the f^{H_S} vertices, $p_{end}^{H_T}$ belongs to the $p_{start}^{H_T}$ opposite edge and no additional intersection points exist.
- Case 10: $p_{start}^{H_T}$ coincides with one of the f^{H_S} vertices, $p_{end}^{H_T}$ is inside of f^{H_S} and no additional intersection points exist.
- Case 11: $p_{start}^{H_T}$ coincides with one of the f^{H_S} vertices, $p_{end}^{H_T}$ is outside of f^{H_S} and the inserted edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ lies on the edge where $p_{start}^{H_T}$ is situated.
- Case 12: $p_{start}^{H_T}$ coincides with one of the f^{H_S} vertices and $p_{end}^{H_T}$ coincides with another f^{H_S} vertex.
- Case 13: $p_{start}^{H_T}$ coincides with one of the f^{H_S} vertices and $p_{end}^{H_T}$ belongs to an adjacent edge of $p_{start}^{H_T}$.
- Case 14: $p_{start}^{H_T}$ is inside of triangle f^{H_S} , $p_{end}^{H_T}$ is outside of f^{H_S} and at least one intersection point p_I is obtained.
- Case 15: $p_{start}^{H_T}$ is inside of triangle f^{H_S} , $p_{end}^{H_T}$ belongs to an edge of f^{H_S} and no additional intersection points exist.
- Case 16: $p_{start}^{H_T}$ and $p_{end}^{H_T}$ are inside of triangle f^{H_S} and no additional intersection points exist.
- Case 17: $p_{start}^{H_T}$ is inside of triangle f^{H_S} , $p_{end}^{H_T}$ is outside of f^{H_S} and the inserted edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ crosses through a vertex of triangle f^{H_S} .
- Case 18: $p_{start}^{H_T}$ is inside of triangle f^{H_S} , $p_{end}^{H_T}$ coincides with one of the f^{H_S} vertices and no additional intersection points exist.

In the example from Figure VI.3, for each line segment of $e(p_{start}^{H_T}, p_{end}^{H_T})$ an appropriate case is assigned to.

Based on the three possibilities in which the starting point of an edge can be found (inside triangle, on an edge or coincident with another vertex), three kinds of SMCC (Structures of Minimal Contour Coverage) are defined as illustrated in Figure VI.5:

1. if $p_{start}^{H_T}$ falls on a vertex p^{H_S} , its SMCC is p^{H_S} 's first ring structure on H_S .
2. if $p_{start}^{H_T}$ falls on an edge e^{H_S} , its SMCC is a 4-sided polygon containing e^{H_S} .
3. if $p_{start}^{H_T}$ falls on an triangle f^{H_S} , its SMCC is the triangle f^{H_S} .

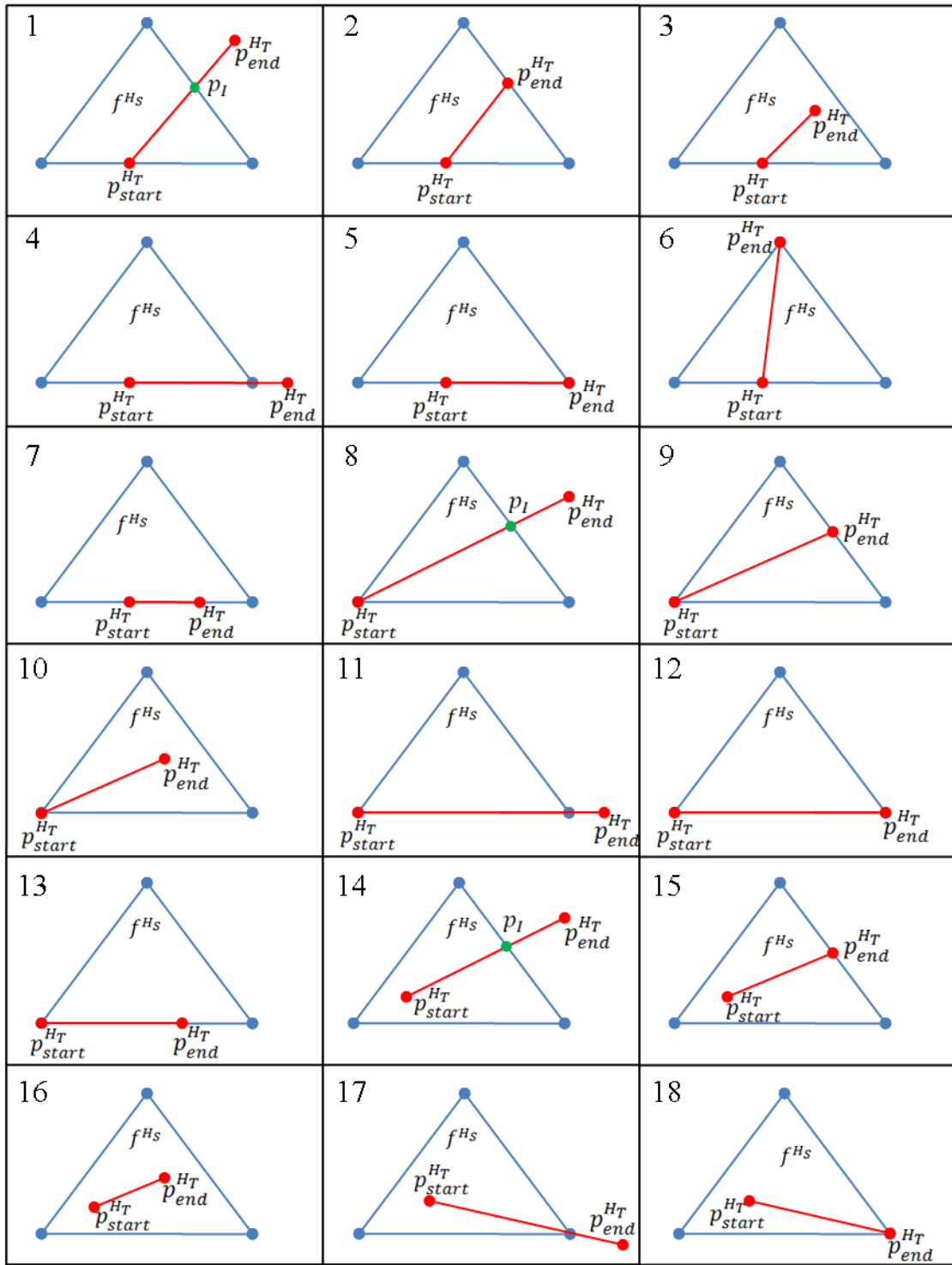


Figure VI.4. Different cases of intersection.

Figure VI.5 shows that each $p_{start}^{H_T}$ is enclosed by its SMCC. In order to compute the intersections, the following two parameters are determined for each edge of the SMCC:

$$\begin{aligned}
 M &= \left(\overrightarrow{p_{start}^{H_T}, p_{end}^{H_T}} \right) \cdot \left(\overrightarrow{p_{start}^{H_T}, p_i^{H_S}} \right) \\
 N &= \left(\overrightarrow{p_{start}^{H_T}, p_{end}^{H_T}} \right) \cdot \left(\overrightarrow{p_{start}^{H_T}, p_{i+1}^{H_S}} \right)
 \end{aligned}
 \tag{VI.7}$$

where $p_i^{H_S}$ and $p_{i+1}^{H_S}$ are two adjacent vertices of $p_{start}^{H_T}$'s SMCC.

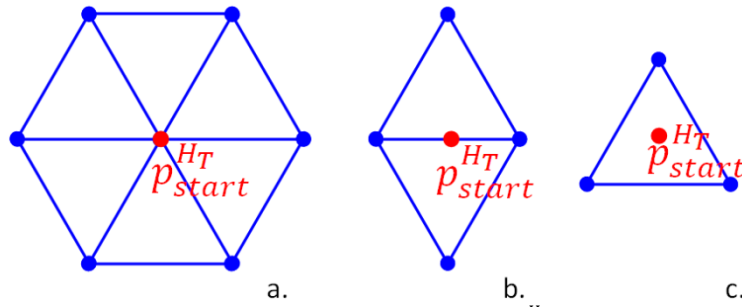


Figure VI.5. The three kinds of SMCC for $p_{start}^{H_T}$ on H_S : (a) first ring; (b) 4-sided polygon; (c) a triangle.

The intersection computation can be evaluated as follows:

- If $M < 0$ and $N > 0$ then edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ intersects $e(p_i^{H_S}, p_{i+1}^{H_S})$;
- If $M = 0$ then edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ cross through vertex $p_i^{H_S}$;
- If $N = 0$ then edge $e(p_{start}^{H_T}, p_{end}^{H_T})$ cross through vertex $p_{i+1}^{H_S}$;
- Else another edge of the SMCC is analyzed.

Once the merging is completed, a non-triangulated planar graph is obtained. In order to retriangulate it and obtain the final metamesh, additional edges must be inserted. This process can be described as follows. For each vertex p_m of the metamesh, the algorithm connects the neighboring points by finding the 1-ring cycles, using the smallest interior angles. For example, in Figure VI.6 vertex p_1 will be connected with p_6 through a new edge and not with p_5 (or other vertices) since the angle $\angle(p_1 p_m p_6)$ is smaller than $\angle(p_1 p_m p_5)$.

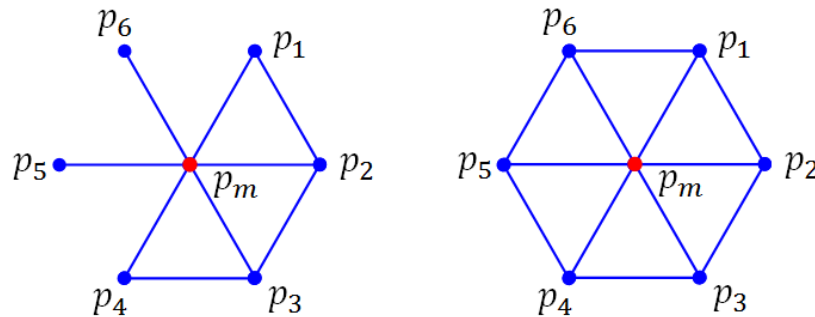


Figure VI.6. First ring neighbors retriangulation.

A different approach that avoids to construct the supermesh using a combination of operations between the topologies of the two models is proposed by Michikawa *et al.* [Mic01]. In this case, the resulted structure that can interpolate between various object shapes is called MIMesh (Multiresolution Interpolation Mesh). The multiresolution interpolation mesh has a semi-regular mesh structure defined by regularly subdividing faces from a base mesh (named base interpolation mesh). A 4-to-1 triangle split scheme is used to subdivide a face into sub-faces. MIMesh triangles are saved in a quad-tree data structure, as illustrated in Figure VI.7. In this structure, the base interpolation mesh triangles are stored in the root node. Each node has links to four child nodes, and each child node stores one of

four sub-faces. The interpolation mesh at an arbitrary subdivision level can be obtained by traversing such a quad tree structure.

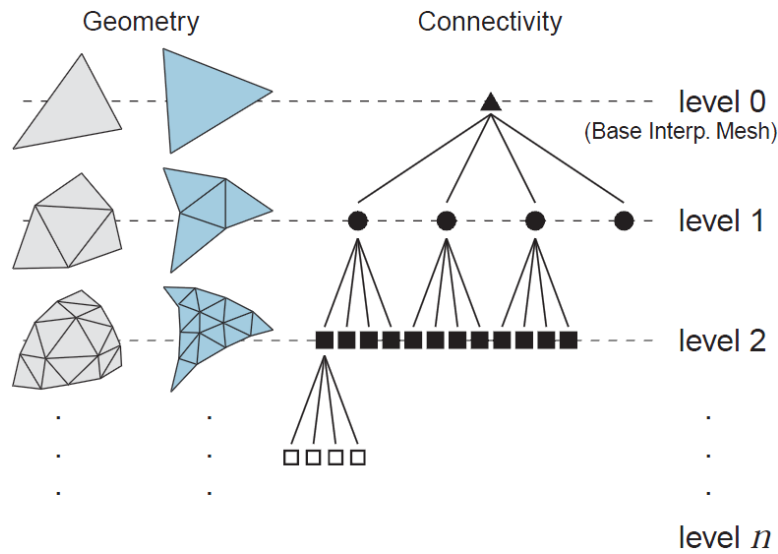


Figure VI.7. Quad-tree structure of MIMesh [Mic01].

In order to better approximate the local geometry of the models without increasing dramatically the number of triangles, the algorithm allows to adaptively modify the number of refinement steps through a local subdivision fitting scheme. An approximation error is defined for each triangle by taking into account the Euclidian distance between its vertices and the original mesh. If the approximation error exceeds a pre-established threshold, the face is split into 4 triangles. However, such a process leads to the apparition of so-called T-vertices, at the level of adjacent triangles with different subdivision levels (Figure VI.8). If a given triangle includes a unique T-vertex, a re-triangulation as the one illustrated in red in Figure VI.8 is applied. If the number of T-vertices is two, the triangulation illustrated in green is applied, followed by a red triangulation to its neighboring triangle.

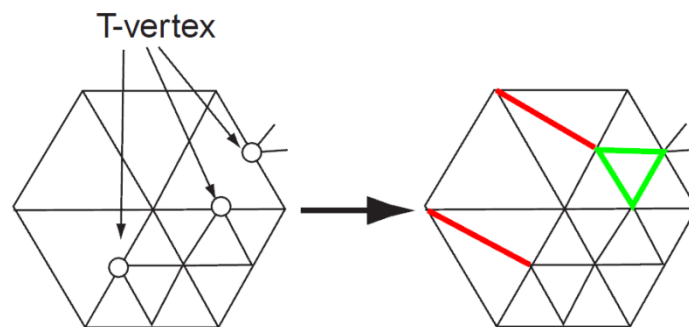


Figure VI.8. Adaptive subdivision scheme to resolve the T-vertices [Mic01].

However, since the vertices and edges of the two input models are not directly used, it is difficult to accurately approximate both the source and target shapes with a fixed vertex set and connectivity. Thus, in such a re-meshing based approach, a large number of subdivision levels is required, which results in a highly complex MIMesh.

A different concept is introduced by Ahn *et al* [Ahn04], which creates in-between meshes based on topology transformation. Given the two input models, their connectivities M^S and

M^T are first converted into some refined meshes, denoted by M'^S and M'^T , which are described by an identical number of vertices.

The converted version (M'^S) of the source mesh topology M^S is constructed in the following way. After the two spherical embeddings of the models H_S and H_T are overlaid, for each parametric vertex p^{H_T} of the target a position into a parametric face f^{H_S} can be found by projection. Then, a vertex p^{H_S} is created at the mapped position and connected to the three vertices of f^{H_S} . This process is performed for all target vertices. Thus a new mesh structure M''^S which contains all the source and target vertices is obtained. Let us note that M''^S has a different connectivity than M^S or M^T (Figure VI.9.b), but can adapt to the M^S 's shape.

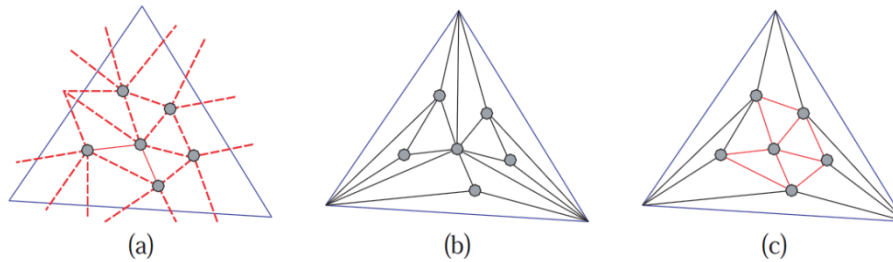


Figure VI.9. Vertices embedding: (a) original configuration of target vertices mapped onto a source triangle; (b) result of simple embedding; (c) enhanced result after edge swaps [Ahn04]

The edges of M''^S connecting the target vertices may differ considerably from the edges in M^T . In order to reduce the differences, a sequence of edge swap operations is applied to M''^S . An edge e''^S of M''^S is swapped only if its endpoints are vertices belonging to the target and the operation reduces the number of intersections between M''^S and M^T on the common embedding. After these operations, the desired converted mesh M'^S is obtained. M^T can be converted to M'^T in the same way. Thus, M'^S and M'^T contain an identical number of vertices equal to $N_S + N_T - N_C$, where N_S and N_T are the number of vertices in source and target models, while N_C is the number of coincident vertices obtained after the mappings are overlaid. M'^S (resp. M'^T) can take the exact shape of the source (resp. target) model.

Next the idea is to construct a minimum edge swap sequence that can transform the connectivity from M'^S to M'^T . This task is accomplished by defining an error metric for each edge swap operation as the shortest distance in 3D between an edge and its swapped version. The edges with the minimum error are treated first. However, an edge e of M'^S is swapped only if the number of intersections of e with M'^T structure decreases.

The drawback of the method comes from the fact that, during the morphing process, the geometric transformation may not be well correlated with the topology. As a consequence, unpleasant visual artifacts may appear. Thus, when an edge swap operation is performed, a pop-up effect may occur if the 3D positions of the initial endpoints of the edge are very dissimilar with the new endpoints. Figure VI.10 illustrates such a pop-up effect.

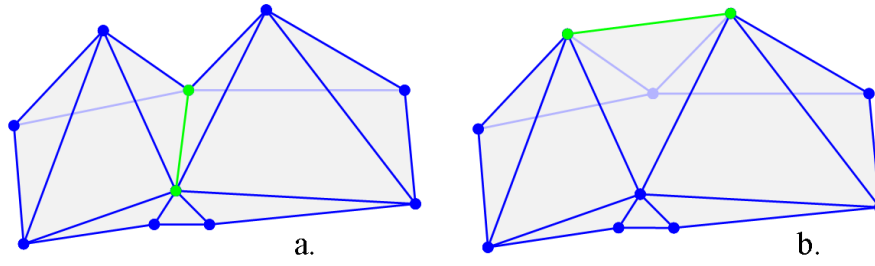


Figure VI.10. Pop-up effect due to edge swap: (a) original mesh; (b) swapped edge

The analysis of the state of the art shows that existing approaches [Ken92], [Kan98], [Ale00], [Urt04], [Lee03] are dealing with the supermesh construction problem by overlapping the two maps of the models, followed by an iterative operation of edge insertion. The metamesh obtained when merging the source and target edges includes all the source and target vertices as well as the new additional intersection points of the edges. However, such an approach proves to highly increase the number of mesh triangles and is very challenging due to numerical instabilities that arise when computing intersections between source and target edges. Thus, a method to create a supermesh characterized by a relatively small number of vertices represents a promising direction of research that we have considered in our work.

In the following section we introduce the proposed method that allows us to obtain a one-to-one correspondence between the shapes of both source and target models, with the help of an adaptive pseudo-mesh construction method.

VI.3. ADAPTIVE PSEUDO-METAMESH CONSTRUCTION

The proposed technique is able to create a so-called pseudo supermesh that avoids performing and tracking edge intersections. In addition, our method reduces drastically the number of vertices normally needed in a supermesh structure. We call our structure pseudo-metamesh since it is not created in the classical manner based on edge intersection, and also it only approximates the two source and target shapes.

We initialize first the supermesh structure with the one of the target parameterization. Then, for each source parametric vertices we establish the supermesh triangle in which it can be projected. In the 2D case (*i.e.*, planar parameterization), this process can be described as follows: Considering the source mapping overlaid on the supermesh structure initialized with the target parameterization, we aim to establish for each source vertex, the target face in which it lies. Considering f^{H_M} a face of the metamesh in the parametric domain H_M described by three vertices $(p_A^{H_M}, p_B^{H_M}, p_C^{H_M})$ and $p_i^{H_S}$ a parametric vertex of the source mesh,

we can determine if $p_i^{H_S}$ lies on f^{H_M} by computing the areas of triangles formed by any two vertices of f^{H_M} and $p_i^{H_S}$.

The area of a triangle $f(p_A, p_B, p_C)$ in plane is given by the following relation:

$$A_f = \frac{1}{2}(x_A(y_B - y_C) + x_B(y_C - y_A) + x_C(y_A - y_B)) \quad (\text{VI.8})$$

which can be further written as:

$$A_f = \frac{1}{2} \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix} \quad (\text{VI.9})$$

Let us note that if the triangle points are specified in counter-clockwise order then the resulting area is positive, whereas the area is negative if the points are specified in clockwise order. This observation makes it possible to decide if a point is situated inside or outside a triangle. Thus, considering the scenario illustrated in Figure VI.11, where the vertex $p_i^{H_S}$ is inside the triangle f^{H_M} , we have the following conditions:

- if the area of triangle $(p_A^{H_M}, p_B^{H_M}, p_i^{H_S})$ is positive, then $p_i^{H_S}$ must be to the left of the edge $(p_A^{H_M}, p_B^{H_M})$.
- if the area of triangle $(p_B^{H_M}, p_C^{H_M}, p_i^{H_S})$ is positive, then $p_i^{H_S}$ must be to the left of the edge $(p_B^{H_M}, p_C^{H_M})$.
- if the area of triangle $(p_C^{H_M}, p_A^{H_M}, p_i^{H_S})$ is positive, then $p_i^{H_S}$ must be to the left of the edge $(p_C^{H_M}, p_A^{H_M})$.

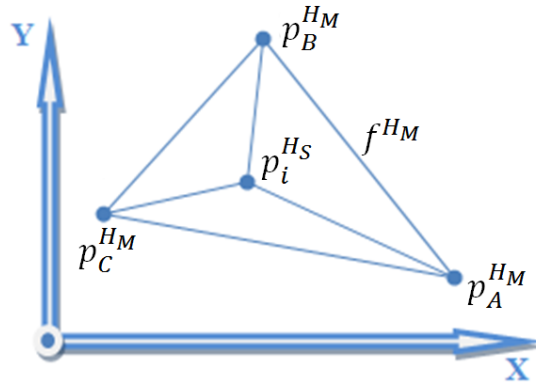


Figure VI.11. Point inside triangle test.

If all the above areas are positives, then the point is inside the considered triangle. Furthermore, if one area is zero, and the other areas are positives, then $p_i^{H_S}$ is on an edge, and if two areas are zero and the other positive, then the vertex is situated on another vertex. Otherwise, $p_i^{H_S}$ is outside the considered triangle.

In the case of spherical parameterization, the problem of determining the metamesh triangle to which a source vertex belongs becomes more complicated. Here, we employ the following ray triangle intersection test in order to establish the location of the source vertices on the

metamesh. We consider the starting point of the ray, the origin of the spherical domain O_S , and its direction specified by a unit vector u defined as $p_i^{H_S} - O_S$. For a given triangle, let n be the associated normal vector, and d a scalar value such that the triangle's plane consists of points x satisfying the following equation:

$$n \cdot x = d \quad (\text{VI.10})$$

If the triangle's vertices are provided in a counterclockwise order, then the vector n is considered to point in an upward direction. Values for n and d can be computed using the following equations:

$$n = (p_B^{H_M} - p_A^{H_M}) \times (p_C^{H_M} - p_A^{H_M}) \quad (\text{VI.11})$$

$$d = n \cdot p_A^{H_M} \quad (\text{VI.12})$$

Let us denote by q the point that intersects the plane defined by the triangle f^{H_M} . With the ray $p_i^{H_S} - O_S$. We first test if point q lies on the triangle plane:

$$d = q \cdot n = O_S \cdot n + ru \cdot n \quad (\text{VI.13})$$

Solving equation (VI.13) for parameter r yields:

$$r = \frac{d - O_S \cdot n}{u \cdot n} \quad (\text{VI.14})$$

If this test finds that $r < 0$, then there is no intersection. By further analysing the sign of either $u \cdot n$ or $d - O_S \cdot n$, we can establish whether the point $p_i^{H_S}$ lies above or below the surface. In order to determine further if the point q is inside or outside f^{H_M} we compute the barycentric coordinates of q with respect to the considered triangle. Thus, the position of the point q is expressed as a convex combination of the vertices $p_A^{H_M}$, $p_B^{H_M}$, $p_C^{H_M}$:

$$q = \alpha \cdot p_A^{H_M} + \beta \cdot p_B^{H_M} + \gamma \cdot p_C^{H_M} \quad (\text{VI.15})$$

where the weights of the convex combination α , β , γ are the barycentric coordinates. Note that $\alpha + \beta + \gamma = 1$, and α , β , γ are all non-negatives. One way to express the barycentric coordinates is in terms of areas of the triangles formed by vertices $p_A^{H_M}$, $p_B^{H_M}$, $p_C^{H_M}$ and $p_i^{H_S}$:

$$\alpha = \frac{|A(\Delta p_B^{H_M}, p_C^{H_M}, p_i^{H_S})|}{|A(\Delta p_A^{H_M}, p_B^{H_M}, p_C^{H_M})|}; \quad \beta = \frac{|A(\Delta p_A^{H_M}, p_C^{H_M}, p_i^{H_S})|}{|A(\Delta p_A^{H_M}, p_B^{H_M}, p_C^{H_M})|}; \quad \gamma = \frac{|A(\Delta p_A^{H_M}, p_B^{H_M}, p_i^{H_S})|}{|A(\Delta p_A^{H_M}, p_B^{H_M}, p_C^{H_M})|} \quad (\text{VI.16})$$

If all the barycentric coordinates α , β , γ belong to $(0, 1)$ interval, then we can affirm that point q lies inside triangle f^{H_M} ($p_A^{H_M}$, $p_B^{H_M}$, $p_C^{H_M}$) and in the same time the point $p_i^{H_S}$ lies on the spherical triangle f^{H_M} . If the point $p_i^{H_S}$ lies on some of vertices $p_A^{H_M}$, $p_B^{H_M}$, $p_C^{H_M}$, then one barycentric coordinates is equal to one and the remaining to zero, whereas if only one barycentric coordinate is null, then the vertex $p_i^{H_S}$ is located on an edge.

Once we determine the face in which a source vertex lies, we split the triangle after a 1-to-4 scheme as illustrated in Figure VI.12. The process is applied for all the source vertices until each triangle in the target mesh includes uniquely a single source vertex.

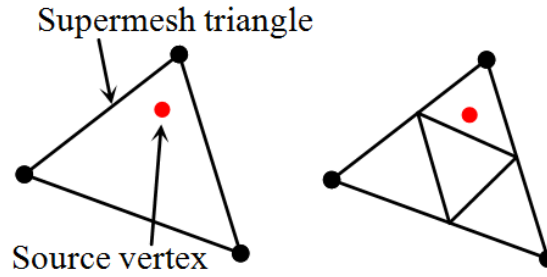


Figure VI.12. 1-to-4 subdivision scheme.

Obviously, the obtained pseudo-metamesh does not have anymore a triangular structure since, after a triangle subdivision the adjacent faces are not triangles anymore. Thus, a mesh retriangulation is required. This task is performed only after all source vertices are used to split the metamesh triangles.

The retriangulation process can be easily accomplished if we store the elements of the metamesh structure in appropriate lists which are updated accordingly after each triangle split. We consider $V\{p_1, \dots, p_{N_V}\}$ and $F\{f_1, \dots, f_{N_F}\}$ the metamesh list of vertices and faces respectively, where N_V and N_F denote the initial numbers of vertices and faces, after the initialization of the metamesh with the target model. For each vertex p_i , the list of adjacent faces $F_{i/adj}$ is known.

In order to illustrate the retriangulation process, let us consider the example in Figure VI.13. The triangle f_k defined by vertices p_A, p_B, p_C (Figure VI.13.a) is split after 1-to-4 subdivision scheme resulting three new faces $f_{N_F+1}, f_{N_F+2}, f_{N_F+3}$, which are added in the F list. The most inner triangle obtained after the subdivision process will take the place of the initial triangle in the F list. Also three new vertices, $p_{N_V+1}, p_{N_V+2}, p_{N_V+3}$, are added in the V list.

The lists of the adjacent faces are updated as follows:

- p_A replace the adjacent face f_k with f_{N_F+1} ;
- p_B replace the adjacent face f_k with f_{N_F+2} ;
- p_C replace the adjacent face f_k with f_{N_F+3} ;
- the new vertex p_{N_V+1} add in the $F_{N_V+1/adj}$ list the following triangles: $f_{N_F+2}, f_k, f_{N_F+3}$;
- the new vertex p_{N_V+2} add in the $F_{N_V+2/adj}$ list the following triangles: $f_{N_F+3}, f_k, f_{N_F+1}$;
- the new vertex p_{N_V+3} add in the $F_{N_V+3/adj}$ list the following triangles: $f_{N_F+1}, f_k, f_{N_F+2}$;

Note that the triangles f_A, f_B, f_C are not added in the lists of adjacent faces of the new vertices.

The list of faces is traversed and the “*triangles*” with more than 3 vertices are detected. In Figure VI.13, we establish, for example, that face f_C is not well defined and the vertex p_{N_V+3} split the edge $e(p_A, p_B)$ in two halves. This is simple established by analyzing the lists of faces adjacent to vertex p_{N_V+3} . If two vertices of a triangle have only one common adjacent face then that face must be retriangulated. In our example, p_A and p_B have only the face f_C adjacent. Thus, f_C will be split into two triangles and the metamesh lists updated accordingly. The two new faces are verified if should be further split.

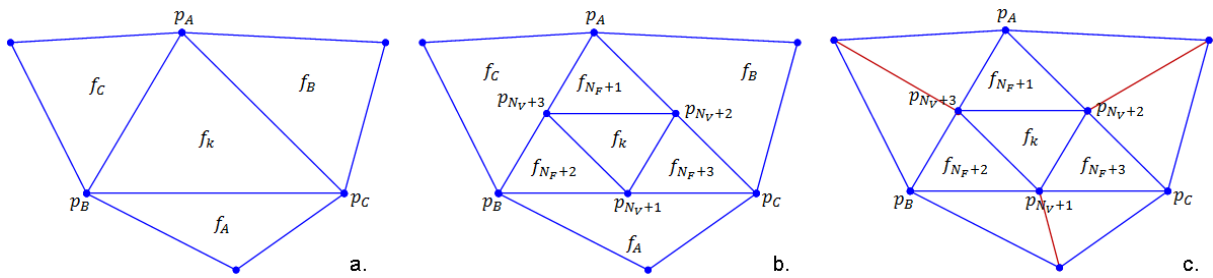


Figure VI.13. Mesh retriangulation: (a) the pseudo-metamesh before the subdivision; (b) the pseudo-metamesh obtained after the 1-to-4 subdivision scheme; (c) retriangulated pseudo-metamesh.

In this manner, the final retriangulated pseudo-metamesh contains only the target vertices and the new vertices obtained by triangle split operations.

The next step aims to establish the 3D positions of these vertices relatively to both source and target shapes. The 3D position of the new vertices relatively to the target shape can be easily established since we know that after each split operation, the new vertices are inserted at the middle of an existing edge. For example, in Figure VI.13, the 3D position of p_{N_V+3} relatively to the target model can be computed as:

$$p_{N_V+3}^T = (p_A^T + p_B^T)/2 \quad (\text{VI.17})$$

The 3D positions of all pseudo metamesh vertices relatively to the source shape can be computed employing a point-in-triangle test as we have presented earlier in this section.

Figure VI.14 illustrates two examples of pseudo-metameshes obtained with the proposed approach. We can observe that the mesh structure remains simple and in the proximity of existing features the supermesh is adaptively remeshed in order to better approximate both original models.

Table VI.1 presents the characteristics of some pseudo-metameshes in terms of number of vertices and triangles compared with the original models. Let us note that in most of the cases the pseudo metamesh number of vertices does not exceed the sum of the source and target vertices, which is quite a remarkable result.

The final step required for obtaining the morphing sequence concerns the interpolation of the geometric positions of the source and target vertices of the pseudo metamesh involved.

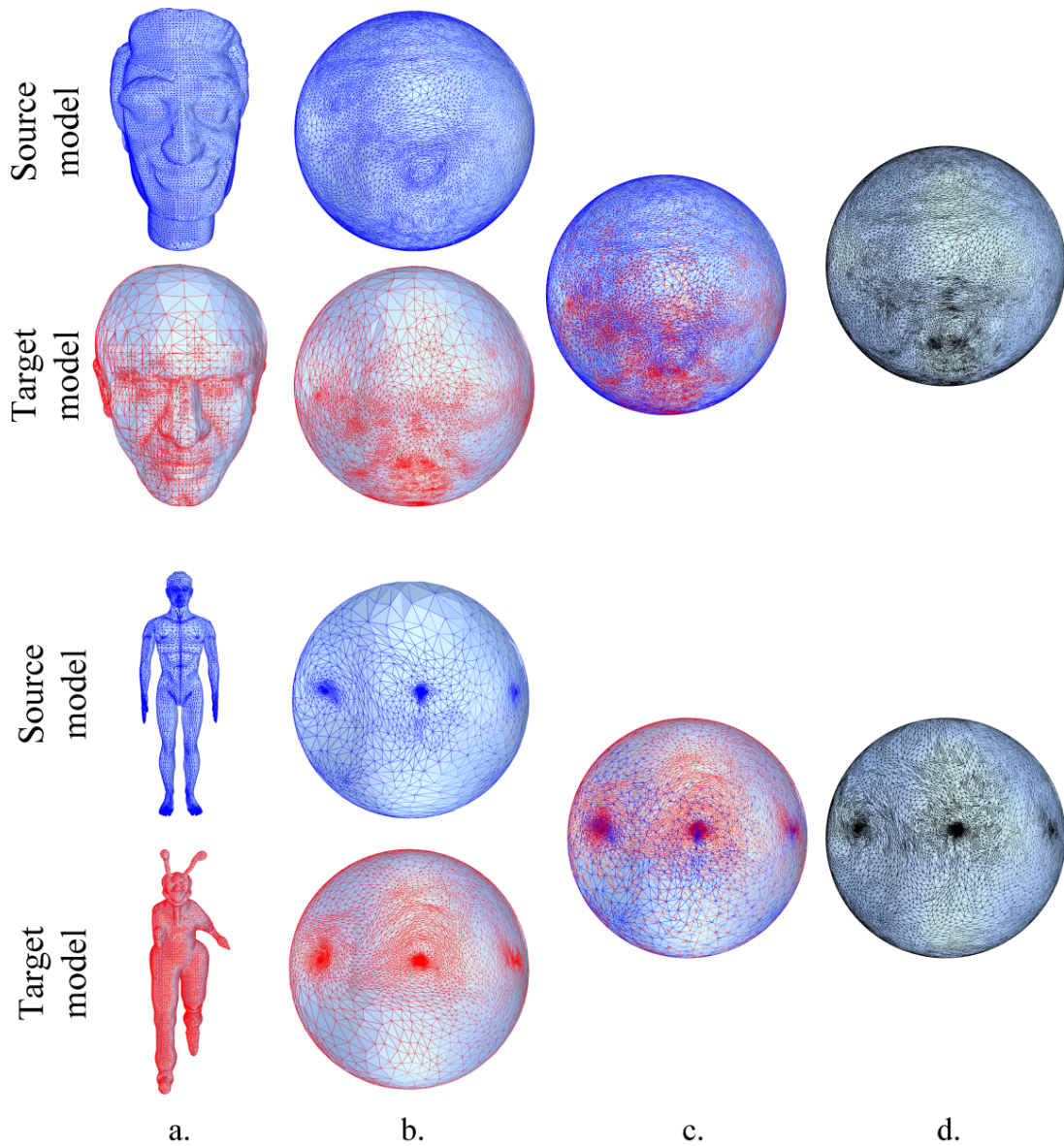


Figure VI.14. Pseudo metameshes: (a) original models; (b) spherical parameterization; (c) overlaid maps; (d) final pseudo metamesh.

Table VI.1. Pseudo metamesh characteristics.

	Model	No. of vertices	No. of triangles		No. of vertices	No. of triangles
Source	Man	14603	29202	Pseudo metamesh	34796	69588
Target	Alien	16267	32530			
Source	Head1	17358	34712	Pseudo metamesh	22467	44930
Target	Head2	7896	15788			
Source	Dino	16996	33988	Pseudo metamesh	31082	62080
Target	Horse	19851	39698			
Source	Cow	11610	23216	Pseudo metamesh	13386	26768
Target	TRex	2832	5660			
Source	Igea	15002	30000	Pseudo metamesh	24789	49574
Target	Head1	17358	34712			

VI.4. MESH INTERPOLATION

The objective of the mesh interpolation step is to determine appropriate trajectories for each vertex connecting the initial position p_i^S to the final position p_i^T in the metamesh.

Used in the majority of morphing applications [Ken92], [Kan00], [Ale00], [Ahn04], [Ath12], the simplest way to interpolate between the initial and the final positions of a vertex is the linear interpolation:

$$p_i^t = p_i^S + t(p_i^T - p_i^S) \quad (\text{VI.18})$$

where p_i^t is the position of the i^{th} vertex of the metamesh at the moment of time t . p_i^S and p_i^T are the extreme vertex positions at the moment $t = 0$, and $t = 1$ respectively.

Due to its simplicity, we have adopted the linear interpolation approach in our work. Figure VI.15 to Figure VI.20 illustrates some examples of metamorphosis between different 3D models obtained using our algorithm. The considered subset of objects consists of 3D closed genus-0 manifold models with various complexities and shapes. All the models are freely available over the Internet and are part of the Princeton and MPEG-7 databases.

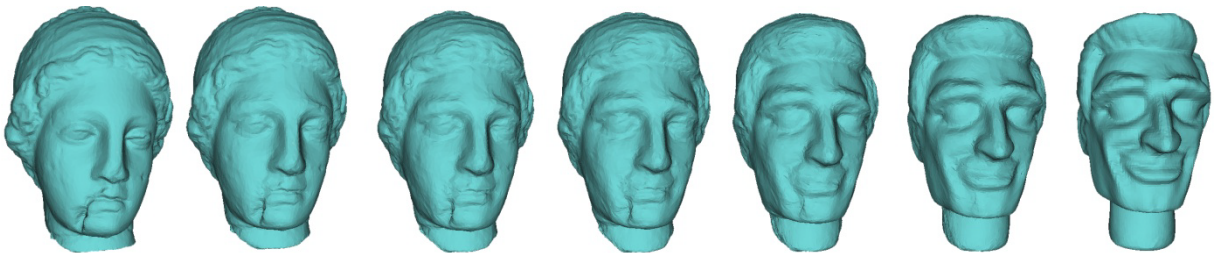


Figure VI.15. Morphing between Igea and Head1 models.

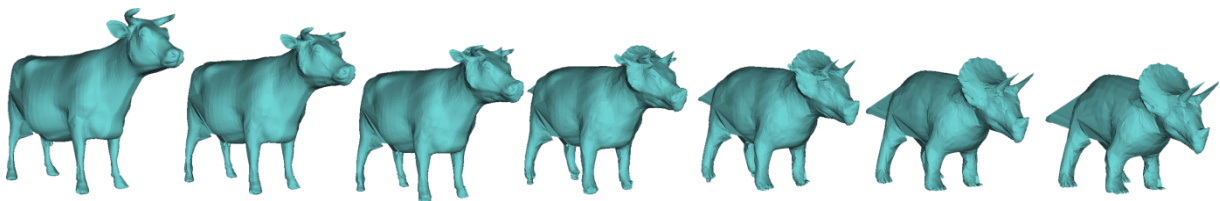


Figure VI.16. Morphing between Cow and TRex models.

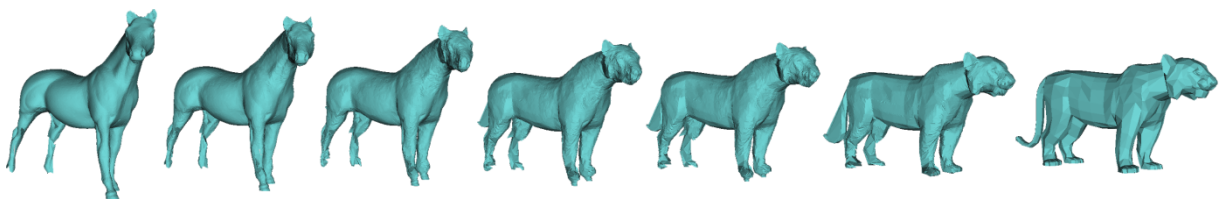


Figure VI.17. Morphing between Horse and Lion models.

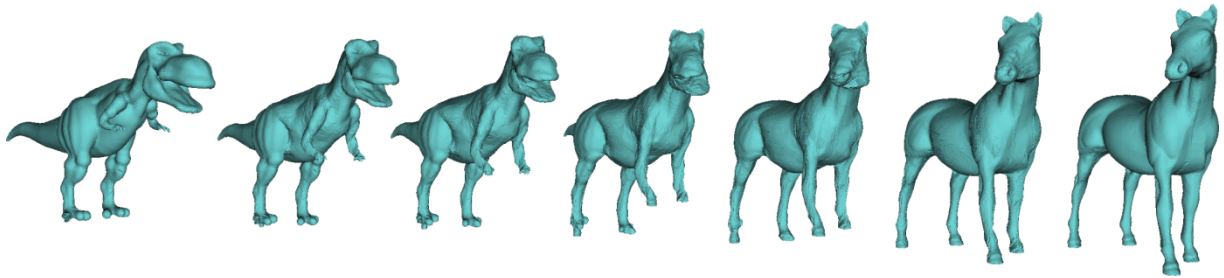


Figure VI.18. Morphing between Dino and Horse models.

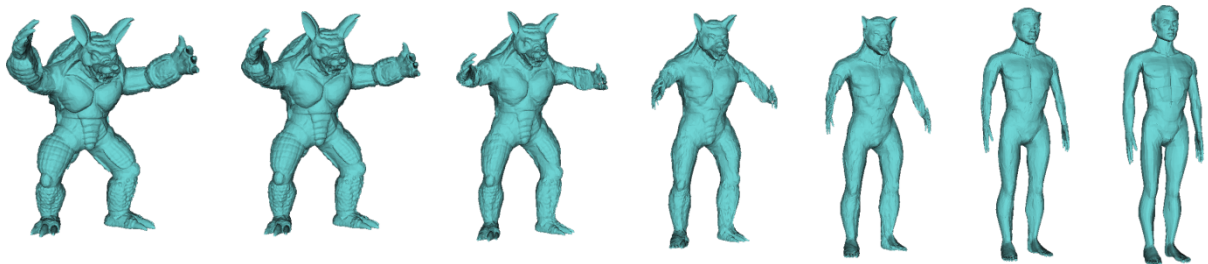


Figure VI.19. Morphing between Armadillo and Man models.

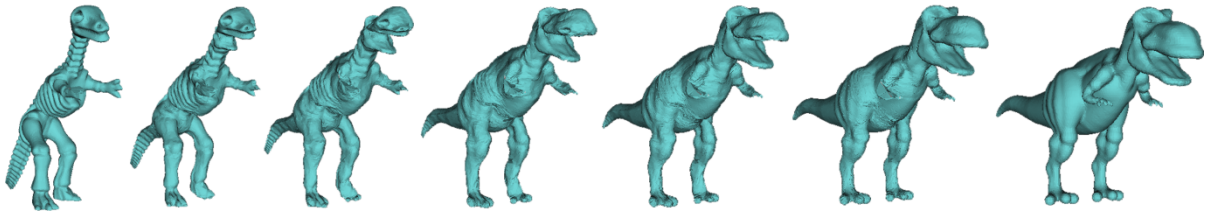


Figure VI.20. Morphing between DinoSkeleton and Dino models.

We can observe that in the majority of cases the resulting morphing sequences ensure a gradual and visually pleasant transition between source and target models. In addition, the pseudo metamesh proposed is able to adapt to both source and target shapes.

However, it can be observed that are some cases when the linear interpolation leads to some minor self-intersections in the model during the morphing sequence. This is visible especially in Figure VI.19 where the hands of the Armadillo model are placed in an entirely different position in space than those of the Man model.

A solution to this problem can be achieved by considering more advanced interpolation methods. Thus, as demonstrated by Alexa in [Ale02], the linear interpolation works well for morphing between 3D models that are similar and oriented in the same direction. For objects with strongly different shapes the linear vertex interpolation may introduce self intersections or some sort of collapsing, which may create disturbing visual effects.

More advanced interpolation techniques are also available, which provide smoother vertex trajectories, but with the cost of higher computational complexity. Usually, they require some additional information, as control vertices for the case of a Bezier interpolation or some tangents information for a Hermite interpolation [Mic01].

A different idea was proposed by Gregory *et al.* in [Gre99] where the user can specify tangent vectors for the vertex path. The modified trajectory is then spread with some falloff to the neighboring vertices. Defining suitable tangent vectors some cases of self-intersection can be avoided.

Besides the methods which interpolate between corresponding vertices, there are the so called *intrinsic interpolation* methods which take into account intrinsic shape parameters. Intrinsic parameters are, e.g., edge lengths or angles between adjacent edges or faces, face areas, etc. By interpolation of such parameters it is possible to force the angles or edges to change monotonically without creating degenerate triangles or generate self intersections. An example of such an intrinsic representation is the edge-angle representation proposed in [Sun97] or the strain field interpolation proposed in [Yan07].

Finally, let us mention the prototype morphing application that allows the user to interactively operate with 3D models and control each step of the morphing process. The intuitive interface permits user to select the correspondent vertices in the two models or to save the processed meshes at any time.

Figure VI.21 shows different views of the user interface layout. The left window display the source model, while the right one displays the target object. The user has the possibility to specify a set of corresponding feature points on both source and target models. Once the computation of the supermesh is completed, this one is displayed on the left part.

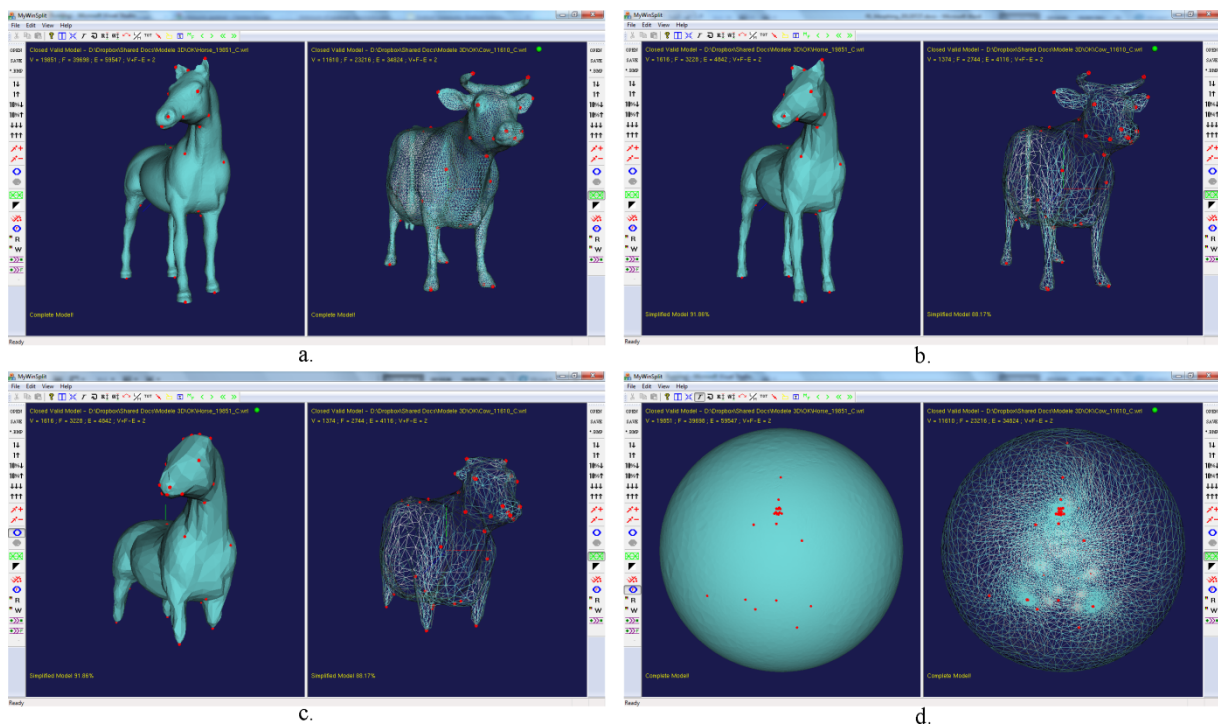


Figure VI.21. Graphical user interface: (a) view with the input models; (b) view during mesh simplification; (c) view during parameterization; (d) view with the final spherical embeddings.

VI.5. CONCLUSIONS

In this chapter we have first presented the state of the art algorithms employed for the construction of a supermesh model. Generally, the supermesh is necessary to interpolate between the source and target shapes and is obtained in the parametric domain by overlapping the mappings of the input models. Most of the approaches proposed in the literature employ an edge intersection scheme between the topology of the two models. Unfortunately, such techniques suffer from numerical instabilities especially when edge intersections are performed in dense regions of vertices. Additionally, the number of vertices increase drastically compared with the input models.

To overcome such limitations, we have introduced in this chapter a new method which build a pseudo metamesh that starts with the target mesh structure and is adaptively refined such that to better approximate both source and target model. Thus, our approach avoids the edge-to-edge intersection process and returns mesh structures with a reduced number of vertices, which is generally inferior the sum of source and target vertices.

The proposed pseudo metamesh has been exploited for morphing purposes, with the help of a linear interpolation technique.

Perspectives of future work mainly concern the issue of smooth interpolation between source and target metamesh vertex positions. More advanced techniques are here required in order to avoid self intersection in the case of meshes with strongly different geometries.

VII. CONCLUSIONS AND FUTURE WORK

In this thesis, we have proposed a novel framework for 3D mesh morphing capable to interpolate between arbitrary genus-0 objects. The technique can be used as an animation method for creation of some special effects or in the design area where two existing shapes are combined in order to obtain new shapes.

Our method is dependent on the objects representation, but it can be easily adapted for various types of descriptions. Based on the analysis provided in Chapter II we have decided to focus, in this thesis, on the mesh boundary representation, since it is very widespread in professional animation tools, easy to store, render and edit.

Chapter III presents the state of the art in both 2D and 3D morphing, highlighting the main principles, advantages and limitations of each family of methods. The morphing framework proposed and considered in our work is also presented here.

In Chapter IV, two different approaches are proposed in order to construct valid parameterizations for both open 3D objects topologically equivalent to a disc and for closed 3D models with sphere-like topology.

Our first approach is a planar parameterization method, which introduces a new barycentric mapping algorithm based on the preservation of the mesh length ratios. The experimental results have proved the superiority of our algorithm compared with state of the art methods by providing low distortions rates in terms of area and lengths especially for complex models. Another major advantage of our method, concerns the bijectivity property, which holds in all cases and ensures valid embeddings for arbitrary open and triangular 3D meshes.

A second contribution concerns a spherical parameterization method, suitable for 3D closed two manifold models. The key point of our method concerns the Gaussian curvature criterion, which makes it possible to iteratively detect salient mesh vertices and to locally flatten them, until a sphere-like surface, adapted to a direct spherical parameterization is obtained. A notable advantage concerns the bijectivity properties that guarantee for any closed 3D mesh, a valid embedding regardless their complexity. The experimental evaluation, carried out on a set of 3D models of various shapes and complexities, has demonstrated a significant improvement in terms of both angle and area distortions.

Another distinctive factor is the complete automatic nature of our planar and spherical parameterization techniques which do not require any human intervention.

Based on the detailed analysis and evaluation of the most important mesh deformation techniques made in Chapter V we have established that the CTPS C_a^2 radial basis function represents the most suitable method for mesh warping purposes. However, we have adjusted this warping technique such that to meet the constraints related to feature alignment of meshes defined in the parametric domain and to produce minimum mesh distortions. Our approach allows to deform the two mesh embeddings until the feature vertices of the two input models are put in correspondence in the parametric domain maintaining a valid spherical mapping through the entire iterative deformation process.

Based on the previous established feature correspondence, in Chapter VI we introduced a novel algorithm for construction of a pseudo-metamesh that avoids the complex process of edge intersections encountered in the state-of-the-art. Additionally, the obtained mesh structure is characterized by a small number of vertices (*i.e.*, inferior to the sum of source and target vertices) and is able to approximate both the source and target shapes. Finally, the proposed pseudo metamesh has been exploited for morphing purposes, with the help of a linear interpolation technique, which leads to the desired transformation sequence between 3D character models while preserving the necessary features.

The entire mesh morphing algorithm was integrated in an interactive application that allows the user to control and visualize all the stages of the morphing process.

Our perspectives of future research concern different axes:

- Morphing objects with different genus is still an issue that has to be resolved. Extension to this problem could require the modification of the entire framework;
- Interpolation of surface attributes as normals, colors, textures;
- Advanced interpolation schemes for obtaining more visually pleasing results, while avoiding the self intersection problem during the morphing especially when the source and target models present strongly different geometric and topological characteristics;
- Specify a fully automatic morphing method or at least minimize the required user interaction.

List of publications

- [1]. B. Mocuana, T. Zaharia, "A Complete Framework for 3D Mesh Morphing", 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI 2012) – Accepted for publication.
- [2]. B. Mocuana, T. Zaharia, "A Pseudo Metamesh Approach for 3D Mesh Morphing", IEEE International Conference on Consumer Electronics (*ICCE*), Las Vegas, Nevada – Accepted for publication.
- [3]. B. Mocuana, T. Zaharia, "Direct spherical parameterization of 3D triangular meshes using local flattening operations", 7th International Symposium on Visual Computing, ISVC-2011, Part I, LNCS 6938, pp. 611–622, Las Vegas, Nevada, USA, 2011 (BDI).
- [4]. B. Mocuana and T. Zaharia, "Length ratio preserving parameterization of triangular 3d meshes", 3rd International Conference on Future Computer and Communication – ICFCC 2011, ISBN: 978-0-7918-5971-1, pp. 77-82, Iasi, Romania, 3-5 June 2011(ISI).
- [5]. B. Mocuana, T. Zaharia, "Direct spherical parameterization based on surface curvature", Workshop on Digital Media and Digital Content Management 2011, ISBN: 978-0-7695-4413-7, pp. 266-269, Hangzhou, Zhejiang China, May 15-16, 2011(IEEE).

REFERENCES

- [Ahn02] M. Ahn and S. Lee, "Mesh metamorphosis with topology transformations", 10th Pacific Conference on Computer Graphics and Applications, pp. 481- 482, 2002
- [Ahn04] M. Ahn, S. Lee and H. Seidel, "Connectivity transformation for mesh metamorphosis". In SGP '04: Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 75–82, New York, USA, 2004.
- [Ale00] M. Alexa, "Merging polyhedral shapes with scattered features", The Visual Computer, vol. 16, pp. 26-37, 2000.
- [Ale01] M. Alexa, "Local control for mesh morphing", Proceedings of Shape Modeling International, pp. 209-215, 2001.
- [Ale02] M. Alexa, "Recent advances in mesh morphing", In Computer Graphics Forum, vol. 21, no. 2, pp. 173-196, 2002.
- [Ale03] M. Alexa, "Differential coordinates for local mesh morphing and deformation", The Visual Computer, vol. 19, no. 3, pp. 105-114, 2003.
- [Ali05] Alias, "Maya 7 Documentation". <http://www.alias.com/>, 2005.
- [All91] R. E. Allen, "The concise Oxford dictionary of current english", Oxford, Clarendon, 1991.
- [All05] P. Alliez, C. Gotsman, "Recent advances in compression of 3D meshes", Advances in Multiresolution for Geometric Modelling, pp. 3 -26, 2005.
- [Ara95] N. Arad and D. Reisfeld, "Image Warping Using few Anchor Points and Radial Basis Functions". Computer Graphics Forum, vol.14, no.1, pp.23-29, 1995.
- [Asi05] A. Asirvatham, E. Praun, and H. Hoppe, "Consistent spherical parameterization," Proceedings International Conference on Computational Science, pp.265–272, 2005.
- [Ath10] T. Athanasiadis, I. Fudos, C. Nikou and V. Stamati, "Feature-based 3D morphing based on geometrically constrained sphere mapping optimization", 25th ACM Symposium on Applied Computing (SAC'10), Sierre, Switzerland, pp.1258-1265, 22-26 March 2010.
- [Ath12] T. Athanasiadis, I. Fudos, C. Nikou, and V. Stamati, "Feature-based 3D morphing based on geometrically constrained spherical parameterization". Computer Aided Geometry Description, vol. 29, pp. 2-17, January 2012.
- [Bai10] Y. Bai, B. Chen, T. Feng, "Improved algorithm for constrained delaunay triangulation mesh generation", 2nd International Conference on Computer Engineering and Technology (ICCET), vol.5, Chengdu, China, pp.156-160, 16-18 April 2010.

- [Bar84] A. BARR, "Global and local deformations of solid primitives". In Computer Graphics (Proceedings of SIGGRAPH 84), vol. 18, pp. 21-30, 1984.
- [Bar07] I. Baran and J. Popovic, "Automatic rigging and animation of 3D characters", ACM Transaction on Graphics, vol. 26, no. 3, pp. 1-8, 2007.
- [Bei92] T. Beier and S. Neely, "Feature-based image metamorphosis", SIGGRAPH'92, Computer Graphics, pp. 35-42, 1992.
- [Ben08] M. Ben-Chen, C. Gotsman, G. Bunin, "Conformal flattening by curvature prescription and metric scaling", Computer Graphics Forum, vol. 27, no. 2, pp. 449-458, 2008.
- [Bie02] H. Biermann, I. Martin, F. Bernardini, and D. Zorin, "Cut-and-paste editing of multiresolution subdivision surfaces," ACM Trans. Graphics, vol. 21, no. 3, pp. 312-321, 2002.
- [Bir04] H. Birkholz, "Shape preserving parametrization of genus 0 surfaces", In Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision WSCG 2004, pp. 57-64, 2004.
- [Bob04] A. I. Bobenko and B. A. Springborn, "Variational principles for circle patterns and Koebe's theorem", Trans. Amer. Math. Soc, pp. 659-689, 2004.
- [Boe07] A. Boer, M.S. Schoot and H. Bijl, "Mesh deformation based on radial basis function interpolation", Computers & Structures, vol. 85, pp. 784-795, 2007.
- [Cao10] J. J. Cao, Z. X. Su, X. P. Liu and H. C. Bi, "Measured boundary parameterization based on Poisson's equation", Journal of Zhejiang University Science, vol.11(3), pp. 187-198, 2010.
- [Che95] M. Chen, M. W. Jones and P. Townsend, "Methods for Volume Metamorphosis", In Image Processing for Broadcast and Video Production, Springer-Verlag, London, 1995.
- [Che99] M. Chen, M. W. Jones and P. Townsend, "Volume distortion and morphing using disk fields". Computers and Graphics, vol. 20(4), pp. 567-575, 1999.
- [Chu97] F. R. K. Chung, "Spectral Graph Theory", American Mathematical Society, 1997.
- [Cla04] U. Clarenz, N. Litke and M. Rumpf, "Axioms and variational problems in surface parameterization", Computer Aided Geometric Design, vol.21, pp. 727-749, 2004.
- [Cor05] F. Cordier and N. Magnenat-Thalmann, "A data-driven approach for real-time clothes simulation," Computer Graphics Forum, pp. 173-183, 2005.
- [Deg03] P. Degener, J. Meseth and R. Klein, "An Adaptable Surface Parameterization Method", Proceedings 12th International Meshing Roundtable IMR '03, pp. 201-213, 2003.
- [Del34] B. Delaunay, "Sur la sphère vide", Bulletin of the Academy of Sciences of the U.S.S.R., Classe des Sciences Mathématiques et Naturelle vol. 7, no.6, pp. 793-800, 1934.
- [Des02] M. Desbrun, M. Meyer, P. Alliez, "Intrinsic Parameterizations of Surface Meshes". Computer Graphics Forum, vol. 21, pp. 210-218, 2002.
- [Dew04] G. Dewaele, F. Devernay and R. Horaud, "Hand motion from 3d point trajectories

- and a smooth surface model". In European Conference on Computer Vision 2004.
- [Dij59] E. W. Dijkstra, "A note on two problems in connexion with graphs". *Numerische Mathematik* vol. 1, pp. 269–271, 1959.
- [Dom10] A. Dainoff and A. Tannenbaum, "Texture Mapping via Optimal Mass Transport", *IEEE Transactions on Visualization and Computer Graphics*, vol.16(3), pp. 419-33, May 2010.
- [Dyn89] N. Dyn, "Interpolation and approximation by radial and related function", In C.K. Chui, L.L. Schumaker and J. D. Ward editors, *Approximation Theory VI*, vol. 1, pp. 211-234, Academic Press, 1989.
- [Eck95] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle, "Multiresolution analysis of arbitrary meshes", *Proceedings of SIGGRAPH*, pp. 173–182, 1995.
- [Edg03] J. Edge and S. Maddock, "Image-based talking heads using radial basis functions", *Theory and Practice of Computer Graphics*, University of Birmingham, UK, pp. 74-79, 2003.
- [Efr11] A. Efros, "Image Warping and Morphing", CMU, Fall 2011 presentation; http://graphics.cs.cmu.edu/courses/15-463/2011_fall/Lectures/morphing.pdf, 2011.
- [Flo97] M. S. Floater, "Parametrization and smooth approximation of surface triangulations", *Computer Aided Geometric Design*, vol. 14(3), pp. 231-250, 1997.
- [Flo02a] M. S. Floater and K. Hormann, "Parameterization of triangulations and unorganized points", In A. Iske, E. Quak, M. S. Floater (Eds.), *Tutorials on Multiresolution in Geometric Modelling, Mathematics and Visualization*, Springer, pp. 287–316, 2002.
- [Flo02b] M. S. Floater, K. Hormann and M. Reimers, "Parameterization of manifold triangulations", *Approximation Theory X: Abstract and Classical Analysis*, pp. 197-209, 2002.
- [Flo03] M. Floater, "Mean value coordinates", In *Computer Aided Geometric Design*, vol.20(1), pp. 19-27, 2003.
- [Flo05] M. S. Floater, K. Hormann, "Surface parameterization: a tutorial and survey", In *Advances in Multiresolution for Geometric Modelling*, Springer Verlag, pp. 157–186, 2005.
- [Fol90] J. D. Foley, A van Dam, S. K. Feiner and J. F. Hughes, "Computer Graphics, Principles and Practice", Addison-Wesley Reading, MA, ISBN: 0-201-12110-7, 1990.
- [Fri05] I. Friedel, P. Schröder, M. Desbrun, "Unconstrained Spherical Parameterization". *ACM SIGGRAPH Technical Sketches*, ACM, New York, USA, 2005.
- [Fuj98] K. Fujimura and M. Makarov, "Foldover-free image warping". *Graphical models and image processing: GMIP*, vol. 60(2), pp.100-111, March 1998.
- [Gar97] M. Garland and P. S. Heckbert, "Surface Simplification Using Quadric Error Metrics", In *24th Annual Conference on Computer Graphics and Interactive*, pp. 209-216, 1997.
- [Gar99] M. Garland, "Multiresolution Modeling: Survey & Future Opportunities". *Eurographics '99, State of the Art Report*. 1999.

- [Gil09] J. Gilbert, "Real Time Morphing Of Polyhedra", Technical report COM3010, December 2009
- [Gom99] J. Gomes, L. Darsa, B. Costa, and L. Velho, "Morphing and Warping of Graphical Objects", Morgan Kaufmann Publishers Inc, 1999.
- [Got03] C. Gotsman, X. Gu and A. Sheffer, "Fundamentals of spherical parameterization for 3d meshes", ACM Transactions on Graphics vol. 22(3) pp. 358–363, July 2003.
- [Gre96] G. Greiner and K. Hormann, "Interpolating and Approximating Scattered 3D Data with Hierarchical Tensor Product Splines". In: A. Le Mehaute, C. Rabut and L. L. Schumaker, Surface Fitting and Multiresolution Methods, pp. 163–172, 1996.
- [Gre98] A. Gregory, A. State, M. Lin, D. Manocha, and M. Livingston, "Feature-based surface decomposition for correspondence and morphing between polyhedra", In Proc. Computer Animation '98, Philadelphia, IEEE CS Press, pp. 64–71, 1998.
- [Gre99] A. Gregory, S. Andrei, C. Ming, M. Dinesh and A. Mark, "Interactive surface decomposition for polyhedral morphing", The Visual Computer Journal, vol. 15, pp. 453-470, 1999.
- [Gu03] X. Gu , S. T. Yau, "Global conformal surface parameterization", Proceedings of the 2003 Eurographics ACM SIGGRAPH symposium on geometry processing, Aachen, Germany, 23-25 June 2003.
- [Gus00] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder, "Normal meshes". In Computer Graphics (Proc. SIGGRAPH2000), pp. 95–102. ACM Press, New York, 2000.
- [Gus02] I. Guskov, A. Khodakovsky, P. Schröder and W. Sweldens, "Hybrid meshes: multiresolution using regular and irregular refinement". In Proceedings Symposium on Computational geometry, pp. 264–272, 2002.
- [Hak00] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro and M. Halle, "Conformal Surface Parameterization for Texture Mapping", IEEE Transactions on Visualization and Computer Graphics, vol. 6, no. 2, pp. 181-189, April 2000.
- [He94] T. He, S. Wang and A. Kaufman, "Wavelet-based volume morphing", In Proceedings of IEEE Visualization' 94, Washington, D.C., pp. 85-92, 1994.
- [Her90] H. Edelsbrunner and E. P. Mücke, "Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms". ACM Trans. on Graphics, vol. 9, pp. 66-104, 1990.
- [Hop93] H. Hoppe, "Mesh Optimization", In Proceedings of ACM SIGGRAPH, pg. 19-26, 1993.
- [Hor99] K. Hormann, G. Greiner and S. Campagna, "Hierarchical parametrization of triangulated surfaces", Vision, Modeling, and Visualization, pp. 219-226, 1999.
- [Hor00] K. Hormann and G. Greiner, "MIPS: An Efficient Global Parametrization Method", Curve and Surface Design, pp. 153-162, 2000.
- [Hu08] J. Hu, X. Liu, Z. Su, X. Shi and F. Liu, "An Efficient Low Stretch Spherical Parameterization", International Conference on Computer Science and Software Engineering, CSSE 2008, Wuhan, China, vol. 2 Software Engineering, 12 - 14 December 2008.

- [Hug92] J. F. Hughes, "Scheduled Fourier Volume Morphing", Proc. SIGGRAPH' 92, In Computer. Graphics, pp. 43-46, 1992.
- [Ise01] M. Isenburg, S. Gumhold and C. Gotsman, "Connectivity Shapes". In Proceedings of IEEE Visualization, San Diego, pp. 135-142, 2001.
- [Jol02] I. T. Jolliffe, "Principal component analysis", 2nd edition, Springer, New York, 2002.
- [Ju05] T. Ju, S. Schaefer and J. Warren, "Mean value coordinates for closed triangular meshes," ACM Trans. Graph., vol. 24, no. 3, pp. 561-566, 2005.
- [Kan97] Y. B. Kang, Y. J. Yu and H. G. Cho, "A New Image Warping Technique using Mesh Patterned Textile", Proceedings of WSCG '97, Plzen, Czech, pp. 202-211, 1997.
- [Kan98] T. Kanai, H. Suzuki, and F. Kimura. "Three-dimensional geometric metamorphosis based on harmonic maps", The Visual Computer, vol. 14(4), pp. 166-176, 1998.
- [Kan00] T. Kanai, H. Suzuki and F. Kimura, "Metamorphosis of Arbitrary Triangular Meshes", IEEE Computer Graphics and Applications, pp. 62-75, 2000
- [Kan06] K. Kaneko, Y. Okada and K. Nijima, "3D Model Generation by Morphing", In Proceedings of the international Conference on Computer Graphics, Imaging and Visualisation, pp. 341-346, July 2006.
- [Kar05] Z. Karni, C. Gotsman and S.J. Gortler, "Free-Boundary Linear Parameterization of 3D Meshes in the Presence of Constraints", Proc. IEEE International Conference Shape Modeling and Applications, pp. 268-277, 2005.
- [Kaz03] M. Kazhdan, B. Chazelle, D. Dobkin, T. Funkhouser and S. Rusinkiewicz, "A Reflective Symmetry Descriptor for 3D Models", Algorithmica, vol. 38, no. 1, pp. 201-225, October 2003.
- [Ken92] J. R. Kent, W. E. Carlson and R. E. Parent, "Shape transformation for polyhedral objects", Computer Graphics, SIGGRAPH Proceedings 26, vol.2, pp. 47-54, 1992.
- [Kha06] L. Kharevych, B. Springborn and P. Schröder, "Discrete conformal mappings via circle patterns", ACM Transactions on Graphics vol. 25(2), pp. 412-438, 2006.
- [Kho03] A. Khodakovsky, N. Litke and P. Schroder, "Globally smooth parameterizations with low distortion". In SIGGRAPH 2003, pp. 350-357, 2003.
- [Kon89] T. Y. Kong and A. Rosenfeld, "Digital topology: Introduction and survey", In Computer Vision, Graphics, and Image Processing, vol. 48, pp. 357-393, 1989.
- [Knu03] P. M. Knupp, "Algebraic Mesh Quality Metrics for Unstructured Initial Meshes", Finite Elements in Analysis and Design, vol.39, pp. 217-241, 2003.
- [Kre59] E. Kreyszig, "Differential Geometry", University of Toronto Press, Toronto, 1959.
- [Lee89] E. T. Y. Lee, "Choosing nodes in parametric curve interpolation", Computer Aided Design, vol. 6, pp. 363-370, 1989.
- [Lee95] S. Lee, K. Y. Chwa, S. Y. Shin and G. Wolberg, "Image Metamorphosis Using Snakes and Free-Form Deformations", Computer Graphics, Proc. SIGGRAPH '95, pp. 439-448, 1995.
- [Lee96] S. Y. Lee, K. Y. Chwa, J. Hahn and S. Y. Shin, "Image Morphing Using Deformation Techniques", The Journal of Visualization and Computer Animation, vol. 7, nr. 1, pp. 3-23, 1995.

- [Lee98] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar and D. Dobkin, "MAPS: Multiresolution Adaptive Parameterization of Surfaces", Computer Graphics, SIGGRAPH '98 Proceedings, pp. 95–104, 1998.
- [Lee99] A. Lee, D. Dobkin, W. Sweldens and P. Schröder, "Multiresolution Mesh Morphing", Proceedings of SIGGRAPH 99, pp. 343-350, August 1999.
- [Lee02] Y. Lee, H. Kim and S. Lee, "Mesh parameterization with a virtual boundary", In Computers and Graphics, pp. 677–686, 2002.
- [Lee03] T.-Y. Lee and P. H. Huang, "Fast and intuitive metamorphosis of 3d polyhedral models using SMCC mesh merging scheme", IEEE Trans. Visualization Computer Graphics, vol. 9(1), pp. 85–98, 2003.
- [Lee05] H. Lee, Y. Tong and M. Desbrun, "Geodesics-Based One-to-One Parameterization of 3D Triangle Meshes", IEEE Multimedia, vol. 12, no.1 January 2005.
- [Ler95] A. Lerios, C. Garfinkle and M. Levoy, "Feature-Based Volume Metamorphosis", In Computer Graphics Proceedings, Annual Conference Series ACM SIGGRAPH, Los Angeles, CA, pp. 449-456, 6-11 August 1995.
- [Lev02] B. Levy and N. Petitjean, "Least squares conformal maps for automatic texture atlas generation". In: Proceedings of SIGGRAPH, pp. 362–371, 2002.
- [Li07] H. Li and R. Hartley, "Conformal spherical representation of 3D genus-zero meshes", Pattern Recognition, vol.40, no.10, pp. 2742-2753, October 2007.
- [Lie04] A. Lieutier, "Any open bounded subset of R^n has the same homotopy type than its medial axis", Computer-Aided Design, vol. 36(11), pp. 1029–1046, 2004.
- [Lin05a] C. H. Lin and T. Lee, "Metamorphosis of 3D polyhedral models using progressive connectivity transformations", IEEE Transaction on visualization and computer graphics, vol. 11(1), pp. 2-12, 2005
- [Lin05b] C. H. Lin, T. Y. Lee, H. K. Chu, and C. Y. Yao, "Progressive mesh metamorphosis," Journal of Computer Animation and Virtual Worlds, vol. 16, pp. 487-498, 2005.
- [Liu08] L. Liu, L. Zhang, Y. Xu, C. Gotsman and S. J. Gortler, "A Local/Global Approach to Mesh Parameterization", Proceedings of Eurographics Symposium on Geometry Processing, Computer Graphics Forum, Copenhagen, vol. 27(5), pp. 1495-1504, 2-4 July 2008.
- [Mac96] R. MacCracken and K. I. Joy, "Free-form deformations with lattices of arbitrary topology," in Proceedings of ACM SIGGRAPH 96, pp. 181–188, 1996.
- [Mai93] J. Maillot, H. Yahia, A. Verroust, "Interactive texture mapping", Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 27-34, September 1993.
- [Man99] A. P. Mangan, R. T. Whitaker, "Partitioning 3D Surface Meshes Using Watershed Segmentation", IEEE Transactions on Visualization and Computer Graphics, vol. 5, no. 4, pp. 308-321, 1999.
- [Mey03] M. Meyer, M. Desbrun, P. Schroder, A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds". In Visualization and Mathematics III, Hege, H.-C., Polthier, (Eds.). Springer-Verlag, Heidelberg, pp. 35–57, 2003.
- [Mic01] T. Michikawa, T. Kanai, M. Fujita and H. Chiyokura, "Multiresolution interpolation

- meshes". In 9th Pacific Conference on Computer Graphics and Applications, IEEE. pp. 60–69. October 2001.
- [Mos05] Mosek software - Constrained quadratic minimization software. Version 3.1r42. <http://www.mosek.com/>.
- [Nin09] S. Ningping, S. Tanaka and S. Wenling, "An Alternative Algorithm of Triangulation of Polygons with Holes", Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 202-205, 12-14 Septembre 2009.
- [Nis93] T. Nishita, K.Fujii and E.Nakamae, "Metamorphosis using Bezier clipping.", In Proceedings of the First Pacific Conference on Computer Graphics and Applications, pp. 162-173, Seoul, Korea, 1993.
- [Pay92] B. A. Payne and A. W. Toga, "Distance Field Manipulation of Surface Models", IEEE Computer Graphics and Applications, vol. 12(1), pp. 65-71, 1992.
- [Pen55] R. Penrose, "A Generalized Inverse for Matrices". Proc. Cambridge Phil. Soc. 51, pp. 406-413, 1955.
- [Pie01] L. A. Piegl and W. Tiller, "Parametrization for surface fitting in reverse engineering", Computer-Aided Design, vol. 33, Issue 8, pg. 593-603, July 2001.
- [Pie10] N. Pietroni, M. Tarini and P. Cignoni, "Almost isometric mesh parameterization through abstract domains", IEEE Transactions on Visualization and Computer Graphics, vol. 16(4), pp. 621–635, July 2010.
- [Pin93] U. Pinkall and K. Polthier, "Computing discrete minimal surfaces and their conjugates", Experiment. Mathematics, vol.2, pp. 15-36, 1993.
- [Pip00] D. Piponi and G. Borshukov, "Seamless texture mapping of subdivision surfaces by model pelting and texture blending", In ACM SIGGRAPH 2000, pp. 471–478, 2000.
- [Pra01] E. Praun, W. Sweldens and P. Schröder, "Consistent mesh parameterizations", Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 179-184, August 2001.
- [Pra03] E. Praun, H. Hoppe, "Spherical parametrization and remeshing", ACM Transactions on Graphics, vol. 22, no. 3, July 2003.
- [Pre94] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, "Numerical recipes in C: the art of scientific computing". 2nd edition, Cambridge University Press, 1994.
- [Pre02] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, "Numerical Recipes in C++: The Art of Scientific Computing", 2nd Edition. Cambridge University Press, Cambridge, UK, 1002pp, 2002.
- [Qiu09] Y. Qiu, Y. Wang, X. Liu, and W. Mio, "Spherical representations of shape using parametrizations with minimal distortion", Proceedings of the Sixth IEEE international conference on symposium on biomedical imaging: from Nano to Macro, Boston, Massachusetts, USA , pp. 674-677, June 2009.
- [Rah07] M. T. Rahman, M. A. Al-Amin, J. B. Bakkre, A. R. Chowdhury, M. A. Bhuiyan, "A novel approach of image morphing based on pixel transformation", Computer and information technology, pp.1-5, 2007.
- [Ros94] J. Rossignac and A. Kaul, "AGRELS and BIPs: Metamorphosis as A Bézier Curve in the Space of Polyhedra", Computer Graphics Forum Eurographics, Oslo,

- Blackwell, vol. 13(3), pp. 179-184, 1994.
- [Rup94a] D. Ruprecht and H. Müller, "Deformed Cross-Dissolves for Image Interpolation in Scientific Visualization", *The Journal of Visualization and Computer Animation*, vol. 5, pp.167-181, 1994.
- [Rup94b] D. Ruprecht, R. Nagel and H. Müller, "Spatial free form deformation with scattered data interpolation methods". Research Report: 539, Department of Computer Science, University of Dortmund, Germany, 1994
- [Sab05] S. Saba, I. Yavneh, C. Gotsman and A. Sheffer, "Practical spherical embedding of manifold triangle meshes", *Proceedings of the International Conference on Shape Modeling and Applications*, pp. 258-267, 13-17 June 2005.
- [Sah96] P. K. Saha, B. B. Chaudhuri, "3D Digital Topology under Binary Transformation with Applications", *Computer Vision and Image Understanding*, vol. 63(3), pp. 418–429, 1996.
- [San01] P.V. Sander, J. Snyder, S.J. Gortler and H. Hoppe, "Texture mapping progressive meshes", In *Siggraph*. ACM Press, pp. 409–416, 2001.
- [San03] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder and H. Hoppe, "Multi-chart geometry images", *Proceedings of the 2003 Eurographics ACM SIGGRAPH symposium on geometry processing*, Aachen, Germany, 23-25 June 2003.
- [Sch98] D. K. Schneider and S. Martin-Michiellot, "VRML Primer and Tutorial", TECFA, University of Geneva, March 1998.
- [Sed86] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models". In *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, vol. 20, pp.151-160, 1986.
- [Sei96] S. M. Seitz and C. R. Dyer, "View morphing", In *Proc. SIGGRAPH 96*, pp. 21–30, 1996.
- [Sha98] A. Shapiro and A. Tal, "Polygon realization for shape transformation", *The Visual Computer* vol. 14, pp. 429-444, 1998.
- [She01] A. Sheffer and E. Sturler, "Parameterization of faceted surfaces for meshing using angle-based flattening", *Engineering with Computers*, vol. 17(3), pp. 326–337, 2001.
- [She02a] A. Sheffer, "Spanning tree seams for reducing parameterization distortion of triangulated surfaces," in *Shape Modelling International*, pp. 61–66, 2002.
- [She02b] A. Sheffer and J. Hart, "Seamster: Inconspicuous low-distortion texture seam layout", in *IEEE Visualization*, pp. 291–298, 2002.
- [She03] A. Sheffer, C. Gotsman and N. Dyn, "Robust Spherical Parameterization of Triangular Meshes", In *Proceedings of 4th Israel-Korea Binational Workshop on Computer Graphics and Geometric Modeling*, Tel Aviv, pp 94-99, 2003.
- [She04] A. Sheffer and V. Kraevoy, "Pyramid Coordinates for Morphing and Deformation", *Proc. 3D Data Processing, Visualization and Transmission Conference (3DPVT)*, pp. 68-75, 2004.
- [She05] A. Sheffer, B. Lévy, M. Mogilnitsky and A. Bogomyakov, "ABF++: fast and robust angle based flattening", *ACM Transactions on Graphics* vol.24(2), pp. 311–330,

- 2005.
- [Sho85] K. Shoemake, "Animating rotation with quaternion curves". In SIGGRAPH 85 proceedings, pp. 245–254, 1985.
- [Sho03] S. M. Shontz and S. A. Vavasis, "A mesh warping algorithm based on weighted Laplacian smoothing", Proceedings of the Tenth International Meshing Roundtable, Sandia National Laboratories, Santa Fe, pp. 147–158, 2003.
- [Sin98] K. Singh and E. Fiume, "Wires: a geometric deformation technique," in SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 405–414, 1998.
- [Smi06] C. Smith, "On Vertex-Vertex Systems and Their Use in Geometric and Biological Modelling", Ph.D. dissertation, University of Calgary, January 2006.
- [Sor02] O. Sorkine, D. Cohen-Or, R. Goldenthal and D. Lischinski, "Bounded distortion piecewise mesh parametrization", in IEEE Visualization, pp. 355–362, 2002.
- [Sor06] O. Sorkine, State-of-The-Art Report: "Laplacian Mesh Processing", Eurographics appeared in Computer Graphics Forum, vol. 25(4), under the title "Differential Representations for Mesh Processing", 2006.
- [Sta07] V. Stamati and I. Fudos, "A Feature-Based Approach to Re-engineering Objects of Freeform Design by Exploiting Point Cloud Morphology", in Proceedings of SPM 2007: ACM Symposium on Solid and Physical Modeling, Beijing China, pp. 347–353, June 2007.
- [Sun97] Y. M. Sun, W. Wang, F. Y. L. Chin, "Interpolation polyhedral models using intrinsic shape parameters", Journal of Visualization and Computer Animation, vol. 8, pp. 81–96, 1997.
- [Sun07] L. Sungyeol, L. Haeyoung, "Parameterization of 3D Surface Patches by Straightest Distances", International Conference on Computational Science, vol. 2, pp. 73–80, 2007.
- [Sun09] H. Sun, L. Ma, "A New Triangulation Algorithm Based on the Determination of the Polygon's Diagonals", CiSE International Conference on Computational Intelligence and Software Engineering, pp. 1–3, 11–13 Decembre 2009.
- [Tau95] G. Taubin, "A signal processing approach to fair surface design". In Proceedings of ACM SIGGRAPH 1995, ACM Press, pp. 351–358, 1995.
- [Tur92] G. Turk, "Re-tiling polygonal surfaces", In Edwin E. Catmull, editor, ACM Computer Graphics (SIGGRAPH '92 Proceedings), vol. 26, pg. 55–64, July 1992.
- [Tut63] W. T. Tutte, "How to draw a graph", Proceedings on the London Mathematical Society, vol. 3(13), pp. 743–768, 1963.
- [Urt04] R. Urtasun, M. Salzmann and P. Fua, "3D Morphing without user interaction", EPFL Technical report 2004.
- [Van92] H. A. Van der Vorst, "BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems". SIAM J Scientific and Statistical Computing, vol. 13, pp. 631–644, 1992.
- [Wac06] A. Wachter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming", Mathematical

- Programming vol.106, pp. 25–57, 2006.
- [Wei10] E. W. Weisstein, "Genus", From MathWorld - A Wolfram Web Resource retrieved august 2010 from <http://mathworld.wolfram.com/Genus.html>
- [Wol90] G. Wolberg, "Digital image warping". IEEE Computer Society Press, Los Alamitos, California, ISBN 0-8186-8944-7, 1990.
- [Wol98] G. Wolberg, "Image Morphing: A Survey", Visual Computer, vol. 14, pp. 360-372, 1998.
- [Wu05] Y. Wu, Y. He and H. Tian, "Relaxation of spherical parameterization meshes", The Visual Computer vol.21(11), pp. 897-904, 2005.
- [Wu07] H.Y. Wu, C. Pan, Q. Yang, and S. Ma, "Consistent correspondence between arbitrary manifold surfaces". In ICCV, pp. 1–8, 2007.
- [Yan07] H. Yan, S. Hu and R. R. Martin, "3D morphing using strain field interpolation", presented at J. Computer Science Technology, pp.147-155, 2007.
- [Yan08] Y. L. Yang, J. Kim, F. Luo, S. M. Hu and X. Gu, "Optimal surface parameterization using inverse curvature map", IEEE Transactions on Visualization and Computer Graphics, vol. 14, no. 5, pp. 1054–1066, 2008.
- [Yan08b] H. B. Yan, S. Hu, R. R. Martin, Y. L. Yang, "Shape Deformation Using a Skeleton to Drive Simplex Transformations", IEEE Transactions on Visualization and Computer Graphics, vol. 14, no. 3, pp. 693-706, May-June 2008.
- [Yos04] S.Yoshizawa, A. G. Belyaev and H.P. Seidel, "A fast and simple stretch minimizing mesh parameterization", International Conference on Shape Modeling and Applications, Genova, Italy, pp. 200–208, June 2004.
- [Yu03] J. B. Yu, J. H. Chuang, "Consistent mesh parameterizations and its application in mesh morphing", Proc. Computer Graphics Workshop, Hualian, 2003.
- [Zay06] R. Zayer, C. Rössl and H. P. Seidel, "Curvilinear Spherical Parameterization," Proceedings International Conference Shape Modeling and Applications SMI '06, pp. 57-64, 2006.
- [Zel02] S. Zelinka and M. Garland, "Permission grids: practical, error bounded simplification", ACM Transactions on Graphics, vol.21(2), pg. 1–25, April 2002.
- [Zha05] E. Zhang, K. Mischaikow and G. Turk, "Feature-based surface parameterization and texture mapping", In ACM Transactions on Graphics, vol. 24(1), pp. 1–27, 2005.
- [Zhu09] Z. J. Zhu and M. Y. Pang, "Morphing 3D Mesh Models Based on Spherical Parameterization", International Conference on Multimedia Information Networking and Security, pp. 309–313, 2009.
- [Zor97] D. Zorin, P. Schroder and W. Sweldens, "Interactive multiresolution mesh editing". In SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 259–268, New York, USA, 1997.
- [Zxu09] Z. Xu and G. Xu, "Discrete schemes for Gaussian curvature and their convergence". Computers & Mathematics with Applications, vol. 57, pp. 1187-1195, April 2009.

Abstract

Morphing methods are today extensively used in computer graphics to simulate the transformation between two completely different objects or to create new shapes by a combination of other existing shapes. It has a variety of applications ranging from special effects in film industry and other visual arts to medical imaging and scientific purposes.

This Ph.D. thesis specifically deals with the issue of metamorphosis of 3D objects represented as 3D triangular meshes. The objective is to elaborate a complete 3D mesh morphing methodology which can ensure high quality transition sequences, smooth and gradual, consistent with respect to both geometry and topology, and visually pleasant. The various steps involved in the transformation process are developed within this thesis.

Our first contributions concern the two different approaches of parameterization: (1) a new barycentric mapping algorithm based on the preservation of the mesh length ratios, and (2) a spherical parameterization technique, exploiting a Gaussian curvature criterion. The experimental evaluation, carried out on 3D models of various shapes, demonstrated a considerably improvement in terms of mesh distortion for both methods.

In order to align the features of the two input models, we have considered a warping technique based on the CTPS C^2_a radial basis function suitable to deform the models embeddings in the parametric domain maintaining a valid mapping through the entire movement process. We show how this technique has to be adapted in order to warp meshes specified in the parametric domains.

A final contribution consists of a novel algorithm for constructing a pseudo-metamesh that avoids the complex process of edge intersections encountered in the state-of-the-art. The obtained mesh structure is characterized by a small number of vertices and it is able to approximate both the source and target shapes.

The entire mesh morphing framework has been integrated in an interactive application that allows the user to control and visualize all the stages of the morphing process.