Distance Based Heterogeneous Volume Modelling

Mathieu Sanchez

A thesis submitted in partial fulfilment of the requirements of Bournemouth University for the degree of Doctor of Philosophy

May, 2015

Copyright statement

Contents

Table of contents ii
List of figures vii
List of tables x
List of Acronyms xi
Abstract xii
Acknowledgements xiii
Declaration xiv

1 Introduction **1**
1.1 Context 2
1.1.1 Object representations 2
1.1.2 Heterogeneous object models 3
1.1.3 Feature based modelling 4
1.2 Problem Statement 4
1.2.1 Distances and predictability 4
1.2.2 Approximate smooth distance fields 5
1.2.3 Heterogeneous object modelling 5
1.2.3.1 Shape awareness and predictability . . . 6
1.2.3.2 Volumetric interpolation through time . 6
1.2.4 Modelling scalar fields, not point sets 6
1.3 Solution Statement 7
1.4 Structure 8

2 Background **10**
2.1 Object representation 11
2.1.1 Boundary representation 11
2.1.1.1 Parametric surfaces 11

2.1.1.2 Polygonal mesh 14
2.1.2 Volumetric representation 15
2.1.2.1 Voxels 15
2.1.2.2 Implicit surfaces 16
2.1.2.3 Constructive Solid Geometry 17
2.1.2.4 Function representation 19
2.1.3 Discussion on object representations 20
2.2 Heterogeneous objects modelling 20
2.2.1 Surface based heterogeneous object modelling . . 21
2.2.2 Voxel based heterogeneous object modelling . . . 22
2.2.3 Volumetric meshes 24
2.2.4 Parametric volumes 25
2.2.5 Scalar fields based methods 26
2.2.5.1 Constructive methods 27
2.2.5.2 Feature-based heterogeneous modelling . 29
2.2.6 Discussion 32
2.3 Distance fields 33
2.3.1 Definition & terms 34
2.3.1.1 Discrete distance fields 35
2.3.1.2 Approximate distance fields 36
2.3.1.3 Interior distances 37
2.3.2 Representation of boundaries by scalar fields . . . 40
2.3.2.1 Polygonal meshes 41
2.3.2.2 Non-uniform rational B-Spline solids . . 45
2.3.3 Main applications of distance fields 47
2.3.3.1 Collision detection 47
2.3.3.2 Sweeping 48
2.3.3.3 Modelling 49
2.3.3.4 Rendering 49
2.3.4 Conclusions on distance fields 50
2.4 Geometric operations 51
2.4.1 Set theoretic operations in FRep 52
2.4.2 Metamorphosis 59
2.4.3 Microstructures 62

2.5 Conclusions 66

3 Distance based heterogeneous volumetric modelling 68

3.1 General approach 69
3.2 Convolution filtering 74
3.2.1 Problem description 74
3.2.2 Criteria of evaluation 75
3.2.3 Existing approaches 75
3.2.4 Overview 76
3.2.5 Kernel size function 77
3.2.6 Kernel function 78
3.2.7 Selection of parameters 79
3.2.8 Parameter restrictions 82
3.2.9 Numerical evaluation of the convolution integral . 83
3.2.9.1 Sample distribution and weights 84
3.2.9.2 Adaptive quality 87
3.2.10 Summary 88
3.3 Shape Conformal Volumetric Interpolation 89
3.3.1 Problem description 89
3.3.2 Criteria of evaluation 91
3.3.3 Existing approaches 92
3.3.4 Overview 93
3.3.5 Interior distance field for material features 95
3.3.6 Voronoi diagram using the interior distance . . . 98
3.3.7 Material interpolation between features 100
3.3.8 Summary 104
3.4 Space time transfinite interpolation 105
3.4.1 Problem description 105
3.4.2 Criteria of evaluation 106
3.4.3 Existing approaches 106
3.4.4 Overview 108
3.4.5 Single-partition objects 108
3.4.6 Partitioned objects 111
3.4.7 User control 114
3.4.7.1 Shape driven interpolation 114

3.4.7.2 Object influence . . . 116
3.4.8 Summary . . . 118
3.5 Morphological shape generation . . . 118
3.5.1 Problem description . . . 118
3.5.2 Criteria of evaluation . . . 119
3.5.3 Existing approaches . . . 120
3.5.4 Overview . . . 121
3.5.5 User-controlled pairwise metamorphosis . . . 122
3.5.5.1 Morphing functions . . . 125
3.5.5.2 Space mappings . . . 125
3.5.5.3 Weighting functions . . . 128
3.5.5.4 Offsets . . . 129
3.5.6 User-controlled group metamorphosis . . . 129
3.5.7 Summary . . . 131
3.6 Conclusions . . . 131

4 Applications and results 133
4.1 Implementation and user interface . . . 133
4.1.1 Convolution filtering use case . . . 135
4.1.2 Shape conforming volumetric interpolation use case 136
4.1.3 Space time transfinite interpolation use case . . . 137
4.1.4 Morphological shape generation use case . . . 138
4.2 Applications of convolution filtering . . . 138
4.2.1 Evaluation . . . 139
4.2.2 Transfinite interpolation . . . 141
4.2.3 Localized smoothing . . . 143
4.2.4 Blending set operations on distance fields . . . 145
4.2.5 Smooth metamorphosis . . . 145
4.2.6 Smooth offset surface . . . 147
4.3 Shape conforming volumetric interpolation . . . 150
4.3.1 Comparison . . . 150
4.3.1.1 Spiral . . . 151
4.3.1.2 Spring . . . 155
4.3.1.3 Spanner . . . 158
4.3.1.4 Character . . . 161

4.3.2 Microstructures 164
4.4 Space time transfinite interpolation 166
4.4.1 Material properties 167
4.4.2 Displacement and other attributes 168
4.4.3 Microstructures 169
4.5 Morphological shape generation 170
4.5.1 Comparison 172
4.5.2 Chairgenics 172
4.5.3 Printed 3D Zoetrope 176
4.6 Summary 177

5 Conclusion **178**
5.1 Main contributions 179
5.2 Future work 180

References **183**

A Calculating interior distances **206**

B Disconnected components and Voronoi diagrams **208**

C Generation of foam microstructures **210**
C.1 Wall thickness 212
C.2 Roundness 212
C.3 Parameter ranges 213
C.4 Seeds distribution 214

D Algorithms **217**
D.1 Space time transfinite interpolation 217
D.2 Shape conforming volumetric interpolation 218
D.3 Feature-based group metamorphosis 218

E List of publications **223**

List of Figures

2.1 A NURBS patch and a NURBS bounded solid 12
2.2 The Stanford bunny model as a mesh 13
2.3 Subdivision applied to Suzanne model 14
2.4 The Utah teapot model represented with voxels 15
2.5 Two metaballs blended together. Image from Wikipedia 17
2.6 A CSG-tree 18
2.7 FRep model example 19
2.8 A spring with gradient material 28
2.9 Material gap 29
2.10 Adaptively sampled Distance Field of the letter R 37
2.11 Euclidean distances vs Interior distances 38
2.12 Euclidean and Manhattan distances 38
2.13 A comparison of R-functions on more primitives 53
2.14 A comparison of R-functions 54
2.15 Metamorphosis example 59
2.16 Simple microstructures 63
2.17 Microstructure parameter interpolation 63
2.18 Multi-scale microstructures 64

3.1 Blending crease 72
3.2 Gaussian weight function 79
3.3 Bump function 80
3.4 Distance to segment 81
3.5 Plot of g 81
3.6 Plot of g, smaller f_c 81
3.7 Extra zero sets for large convolution region 82
3.8 Sample distribution and filtering 86

3.9 Impact of sampling rate for filtering 86
3.10 Sampling rate control field 87
3.11 Spiral case 1 90
3.12 Spiral case 2 90
3.13 Spiral case 3 91
3.14 Steps for shape conformal volumetric interpolation 94
3.15 Interior distance field of a spanner 97
3.16 A Voronoi diagram of four sites (a) and of two volumes (b) 99
3.17 Voronoi partitioning of a teapot 101
3.18 Adjusting s_{ij} offsets to sources 103
3.19 Space time transfinite interpolation figure 109
3.20 Space time interpolation of colours 111
3.21 Partitioned objects issue 112
3.22 Metamorphosis with time-variant volumetric materials . 113
3.23 Influence of the time-gap parameter 115
3.24 Example of different time gap values 116
3.25 The influence of the balance parameter 117
3.26 Matching features 123
3.27 Feature based, pairwise metamorphosis 125
3.28 Deformation without metamorphosis 126
3.29 The monster model (Buddha/Armadillo) 127

4.1 DAG Hierarchy example 134
4.2 DAG Hierarchy for convolution filtering and attribute view 135
4.3 DAG Hierarchy for shape conforming interpolation . . . 136
4.4 DAG Hierarchy for space time transfinite interpolation . 137
4.5 Morphological shape generation process 138
4.6 Numerical evaluation of the field smoothness, gear 140
4.7 Comparison of various fields with transfinite interpolation 142
4.8 Smoothing filter: knife 144
4.9 Smoothing filter: handle 144
4.10 Blended union using filters 146
4.11 Blended difference using filters 147
4.12 Metamorphosis using filters 148
4.13 Offset on smooth distance fields 149

4.14 Spiral cases 151
4.15 Spiral case 1 152
4.16 Spiral case 2 153
4.17 Spiral case 3 154
4.18 Spring cases 155
4.19 Spring case 1 156
4.20 Spring case 2 157
4.21 Spanner cases 158
4.22 Spanner case 1 159
4.23 Spanner case 2 160
4.24 Charlie poses 161
4.25 Charlie character using transfinite interpolation 162
4.26 Charlie character using transfinite interpolation 162
4.27 Charlie weights using transfinite interpolation 163
4.28 Charlie weights using our method 165
4.29 Controlling microstructure properties 165
4.30 STTI metamorphosis using textures 168
4.31 STTI for displacement 169
4.32 STTI for microstructures 170
4.33 User-controlled group metamorphosis 171
4.34 Metamorphosis comparison 173
4.35 Printed chairs 174
4.36 Formnation chair selection 175
4.37 Printed Zoetrope, by William Copley 176
4.38 Zoetrope prototype, by William Copley 176

5.1 Smooth set-theoretic operations on the elements of a set 182

B.1 Voronoi diagram and disconnected components 209

C.1 Sphere packing 214
C.2 Polymer foam in teapot 215
C.3 Radius driven distribution 215

List of Tables

2.1 A comparison R-functions desirable properties 58

List of Acronyms

AABB Axis Aligned Bounding Box

ADF Adaptively sampled Distance Field

BRep Boundary Representation

BReps Boundary Representations

BVH Bounding Volume Hierarchy

CAD Computer-Aided Design

CSG Constructive Solid Geometry

FMM Fast Marching Methods

FRep Function Representation

GPU Graphics Processing Unit

MVC Mean Value Coordinates

NURBS Non-Uniform Rational B-Spline

OBB Oriented Bounding Box

OBBs Oriented Bounding Boxes

PCA Principal Component Analysis

POLA Principle Of Least Astonishment

RBF Radial Basis Function

SAH Surface Area Heuristic

SDF Signed Distance Field

STTI Space Time Transfinite Interpolation

SARDF Signed Approximate Real Distance Functions

TI Transfinite Interpolation

Abstract

Natural objects, such as bones and watermelons, often have a heterogeneous composition and complex internal structures. Material properties inside the object can change abruptly or gradually, and representing such changes digitally can be problematic. Attribute functions represent physical properties distribution in the volumetric object. Modelling complex attributes within a volume is a complex task. There are several approaches to modelling attributes, but distance functions have gained popularity for heterogeneous object modelling because, in addition to their usefulness, they lead to predictability and intuitiveness.

In this thesis, we consider a unified framework for heterogeneous volume modelling, specifically using distance fields. In particular, we tackle various issues associated with them such as the interpolation of volumetric attributes through time for shape transformation and intuitive and predictable interpolation of attributes inside a shape. To achieve these results, we rely on smooth approximate distance fields and interior distances. This thesis deals with outstanding issues in heterogeneous object modelling, and more specifically in modelling functionally graded materials and structures using different types of distances and approximation thereof. We demonstrate the benefits of heterogeneous volume modelling using smooth approximate distance fields with various applications, such as adaptive microstructures, morphological shape generation, shape driven interpolation of material properties through time and shape conforming interpolation of properties. Distance based modelling of attributes allows us to have a better parametrization of the object volume and design gradient properties across an object. This becomes more important nowadays with the growing interest in rapid prototyping and digital fabrication of heterogeneous objects and can find practical applications in different industries.

Keywords: Scalar fields, heterogeneous volume modelling, signed distance fields, C^1-continuity, volumetric interpolation

Acknowledgements

First of all, I would like to express my gratitude to my amazing supervisory team, Prof. Alexander Pasko, Dr Valery Adzhiev, and Prof. Peter Comninos for their continued support and advice.

I would like to thank my examiners for making suggestions which made this document more accurate and readable.

I wish to express my sincere gratitude to Dr Oleg Fryazinov for the help he provided in mathematics and my academic life, and the people at Uformia AS, Turlif Vilbrandt in particular, for their insight in the world of 3D printing and geometric modelling. I am also grateful and indebted to Pierre-Alain Fayolle for his contributions on convolution filtering and the numerous discussions by emails on various topics.

Special thanks go to European Union Interreg IVA project for the scholarship, which made it possible for me to undertake this research. I would also like to acknowledge with gratitude the kindness and support of Jan Lewis.

I would also like to thank all my colleagues and friends at the NCCA, back home in France, and elsewhere. I would also like to express my gratitude to Min Jiang, Tomas Howard, Shujie Deng, Tuomo Rinne and Shihui Guo, for the amazing time I had during my few years at Bournemouth University.

Last but not least, I would like to thank my brother, Nicolas Sanchez, my father Alberto Sanchez and my mother Monique Klima for their support during all those years.

Declaration

This thesis has been created by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated.

The work related to convolution filtering and its applications has been published in Sanchez *et al.* (2015). I participated in the solution formulation and experimented with various numerical implementation of the filter. I did the original implementation, the 3D version of the convolution filtering and the adaptive sampling method. The applications were generated with my plug-in in Maya.

The materials related to space time transfinite interpolation have been published in Sanchez *et al.* (2014). For this paper, I participated in the formulation of the problem and its algorithmic solution. I worked on the prototype implementation, I added the additional controls to help users control the transformation easily, and the images were generated with my Maya plug-in (except the Banana to watermelon example).

The works regarding shape conforming volumetric interpolation were published in Fryazinov *et al.* (2015). I identified the problem, proposed the solution and implemented in Maya. All the examples were done with my plug-in.

The materials related to morphological shape generation have been published in Sanchez *et al.* (2013). I participated in the problem statement, I proposed the solution, implemented it in Maya, and generated all the results unless stated.

The comparison of the various data structures for distance fields were published in Sanchez *et al.* (2012). The implementation and the data was generated by a program I wrote which runs the various methods and techniques automatically.

The full list of publication is shown in appendix E.

Chapter 1

Introduction

Natural objects and some man-made objects are often heterogeneous in their composition, including complex internal structures composed of diverse materials, and possess variable densities and other physical properties. Properties and internal structures may vary at various magnitudes of scale. Heterogeneous object modelling is the process of describing and defining those properties and structures. This type of modelling has a vast range of applications, including bio-engineering, animation, medical research, multi-material design and fabrication, geological and physical simulations. Heterogeneous object design and fabrication has become more prevalent in the recent years. For instance, modern 3D printing processes allow for heterogeneous printing with various materials and colours. Therefore, heterogeneous object modelling will become more important in the coming years. Heterogeneous objects can be split into two main categories; *composite* objects where the materials change abruptly, such as a knife with a wood handle and a metallic blade, and objects with *functionally graded* materials and structures, where there are no clear boundaries between materials or properties, such as a watermelon. While *composite* objects can be managed by traditional computer aided design software systems, *gradient* materials are more challenging. This thesis deals with outstanding issues in heterogeneous object modelling, and more specifically in modelling functionally graded materials and structures. Properties such as density, porosity, material colour and

temperature differ in physical nature and can be measured at any point in the object. Attribute functions represent physical properties distribution in the volumetric object. Modelling complex attributes within a volume is a complex task. Using distances to the boundaries or object features in modelling is a useful way of parametrizing space (Siu and Tan 2002b; Biswas *et al.* 2004; Fayolle *et al.* 2006). In this thesis, the notion of distance is used to advance the field of heterogeneous object modelling through the use of spatial distance fields, specifically, to model smoothly varying object properties, with respect to the shapes being modelled. The focus is placed on user-control of smooth variation of attributes rather than modelling composite objects.

1.1 Context

1.1.1 Object representations

There are several geometric object representations, each with its own pros and cons. Boundary representations describe an object through a set of primitive surfaces such as triangles or free form surfaces. Boundary representations are efficient and easy to use, however they often contain defects as mentioned in Bøhn and Wozny (1992); Butlin and Stops (1996); Campen *et al.* (2012); Jacobson *et al.* (2013); Rajab *et al.* (2013). Boundary representations are most suitable for homogeneous object modelling. On the contrary, volumetric representations clearly define the interior of an object through either a set of elements or continuous functions. Voxels are convenient to query and operate on thanks to their simple explicit nature, and they are already popular in some fields such as scientific visualisation. However, voxels and volume meshes are resolution dependent and are not exact representations. With 3D printers quality improving every year, memory limitations are already an issue for those representations. The Function Representation (FRep) (Pasko *et al.* 1995) uses a real valued function of point coordinates to define an object, where the sign of the function value at any given point in space indicates the point membership. For the purpose of heterogeneous

object modelling, the function representation seems to be more suitable and robust. The constructive hypervolume framework (Pasko *et al.* 2001) extended the FRep model to support an arbitrary number of attributes which can be defined by scalar functions. Indeed, scalar fields seem to be better suited to represent both low and high frequency properties across the entire volume of the object. However, other object representations can also be used in heterogeneous object modelling.

1.1.2 Heterogeneous object models

Aside from object representation, heterogeneous object modelling is particularly challenging. Providing users with precise and easy control of the attributes throughout the volume can be difficult. In particular, smoothly varying attributes throughout the object are not easily controlled. A number of works focus on providing precise and user-friendly control of volumetric attributes. These works can be split into two major classes (Kou and Tan 2007). *Evaluated models*, such as voxels and volumetric meshes, rely on discretization of the object. They are particularly well suited for simulation-based modelling but they can only approximate the user's intent and do not provide an easy control of the field. The other class, *unevaluated models*, are based on a scalar function of point coordinates that can be evaluated at the given point using its closed form or procedural definition. This approach is more popular for modelling attributes because it can provide precision while remaining concise. In unevaluated models, two non mutually exclusive important concepts are used: constructive methods (such as Kou and Tan (2005) and Pasko *et al.* (2001)) and feature-based methods (such as Siu and Tan (2002b) and Biswas *et al.* (2004)). Tree-based methods allow users to build complex shapes from simple operations on primitive shapes and construct the object attributes alongside its geometry. Feature based modelling, on another hand, are user-friendly and provide the user with a good control of the volumetric attributes.

1.1.3 Feature based modelling

There are numerous techniques to model heterogeneous object based on some features. In the majority of cases, the technique uses a function of the distance to a feature (Kou and Tan 2007) or an approximation of it. Distance fields are popular for heterogeneous object modelling because they allow users to design and control the attributes directly rather than rely on numerical simulations, without requiring a lot of work from the user. *Feature*-based modelling with the transfinite interpolation method is a powerful concept. A material *feature* defines a region of space where the attribute value is known. Material *features* can be any geometric primitives such as points, lines, surfaces or even solids. These features can be defined by the user or automatically detected by some procedures. The transfinite interpolation method (Rvachev *et al.* 2001) can then be used to interpolate in between the material *features* (called material *gaps*). As stated in Biswas *et al.* (2004), the transfinite interpolation behaves more intuitively if distance fields are used. Attributes in this case can be expressed with functions of distances.

1.2 Problem Statement

1.2.1 Distances and predictability

Feature-based modelling relies on distances to achieve satisfying results (Biswas *et al.* 2004). First, almost any feature type can be represented with a distance field, which allows for the model to be abstracted from specific feature types such as curves, surfaces and polygonal meshes. Secondly, the behaviour of these fields provide predictable and gradual change of the attributes. The function representation uses scalar fields to define the object and its surface, but most operations focus on the resulting surface rather than the overall field (Fayolle *et al.* 2008). However, such scalar fields can behave unexpectedly and cause the interpolation to be biased or simply incorrect (non monotonic with respect to distance) if the scalar field does not have distance properties. Distance and signed

distance fields are crucial for heterogeneous volumetric modelling to allow the users to use a wide range of geometric primitives and to keep the system predictable. However, some issues persist, such as degree of continuity and intuitiveness. This thesis deals with the distance-based heterogeneous volumetric modelling and addresses several open research issues in this area.

1.2.2 Approximate smooth distance fields

If an attribute function is based on a distance field, the C^1 discontinuities of the distance function will typically propagate to the attribute function (Biswas *et al.* 2004). While this could be tolerable in some system, it is often undesirable and may even be unacceptable in others. C^1 discontinuities will cause "stress concentrations and undesirable singularities" (Biswas *et al.* 2004), and may perturb or even disqualify some algorithms. For these reasons, the use of approximate smooth distance fields are desirable. Such fields need to preserve the distance field zero-level point set (object boundary surface), and maintain C^1 continuity away from the surface. The problem of the smooth approximate distance field is tackled in this thesis. An exact, shape preserving convolution filter is applied to the distance function to remove C^1 discontinuities away from the surface.

1.2.3 Heterogeneous object modelling

Modelling heterogeneous objects and in particular the gradual changes of their volumetric properties is limited to a few techniques such as the transfinite interpolation and source-based modelling. Yet, those techniques are not always suitable or sufficient. Only a few commercial software tools are available for modelling heterogeneous objects, and often concentrate on numerical simulation rather than design. Those limitations prevent expressing material changes in respect to a shape or to interpolate two material distributions through time for transformation

between two given objects. In this thesis, both these problems are investigated.

1.2.3.1 Shape awareness and predictability

The transfinite interpolation method fails to provide intuitive and predictable results under certain circumstances. This global operation fails to translate accessibility and perceived proximity of points in the object, since the shape of the object is not taken into account. This issue means that the results can be unpredictable, and in some cases it can be difficult to design the attributes for complex shapes. These seemingly simple problems have not yet been solved and show the real absence of available tools and techniques for modelling heterogeneous objects. In this thesis, this issue is tackled using interior distances and Voronoi diagrams to perform an intuitive and predictable interpolation between material features within an object.

1.2.3.2 Volumetric interpolation through time

The interpolation of two volumetric material distributions in time can prove challenging with the set of tools and techniques available to date. When a geometric metamorphosis or a shape deformation happens, volumetric material properties may change. A naive linear interpolation of the attributes is not sufficient, since the shapes have no influence over the interpolation. To overcome the lack of existing techniques, this thesis introduces the concept of space time transfinite interpolation that provides a shape aware solution to perform the interpolation of the volumetric material distributions for objects changing their shape in time.

1.2.4 Modelling scalar fields, not point sets

There are many application areas, which benefit from heterogeneous volumetric modelling. In fact, since the user designs and manipulates volumetric attributes, these attributes can be used to control parameters of

geometric operations. Attribute modelling is concerned with modelling scalar fields, which can also be used for geometric modelling. The implication of this statement is that distance fields and smooth distance fields are important not only for heterogeneous volumetric modelling, but also geometric modelling. Modelling microstructures and internal structure attributes are obvious applications. However, scalar fields can also control operation parameters, which could alter the global shape of an object. In this thesis, a group shape metamorphosis operation is used to adaptively generate an object taking various features from a number of given reference objects.

1.3 Solution Statement

Several problems were identified related to heterogeneous object modelling. This thesis solves a number of problems highlighted above.

First, we propose an approximate smooth distance function from any distance function by using convolution filtering. Convolution filtering preserves the exact zero-level set, and is smooth away from the surface. It can be controlled by the user for smoothness or distance variation. Unlike the works presented in Biswas *et al.* (2004) and Fayolle *et al.* (2006), it can be applied on arbitrary distance function so that any mesh or other Boundary Representation (BRep) objects can be used as features, given that a distance function to this object exists. Convolution filtering will be evaluated for smoothness using edge filters on gradients, and visually evaluated on a number of applications such as blending unions, interpolation of material properties and smooth offsets.

Next, shape conforming volumetric interpolation is proposed to solve the issues of predictability and intuitiveness for the transfinite interpolation (Rvachev *et al.* 2001). The solution revolves around two independent ideas: the use of interior distances for shape awareness, and the use of Voronoi diagrams to control the influence of each feature in relation to each other. This method will be evaluated by comparing it with the conventional transfinite interpolation both visually and numerically. For the

numerical comparison, a curve along the medial axis will be evaluated using both techniques, and the graphs of the weights will be compared and discussed.

Then, for the interpolation of volumetric properties through time, an extension of the transfinite interpolation to space time is proposed. Unlike linear interpolation, space time transfinite interpolation is shape driven. The success of the technique will be evaluated through a number of applications to show that it fulfils its objectives.

Finally, morphological shape generation is investigated as a supporting element to the use of scalar field modelling for shape design. The results of this method are compared with various other methods for metamorphosis. Real applications, which were subsequently exhibited in a leading museum, allowed us to conclude that this approach was practical and had potential to be used in the creative industry.

1.4 Structure

This thesis is structured as outlined below.

In chapter 2, the related works are presented. The chapter is divided in four main parts. First, different object representations in geometric modelling are discussed, with a particular emphasis on their capabilities and limitations for heterogeneous object modelling. Secondly, previous work in heterogeneous object modelling is discussed. This discussion leads to a complete survey of distance fields and their applications. Finally, some particular geometric operations are reviewed. Those operations are either instrumental to heterogeneous object modelling, or valuable operations which may be used in heterogeneous object modelling. This chapter will reveal several unresolved problems in the current literature.

In chapter 3, the theoretical contribution of this work is presented. The general approach of this work is first introduced, where the properties of a heterogeneous object are described by functions of distances and time. A number of problems introduced in the previous chapter will

be investigated. First, a shape preserving, smooth approximate distance field will be introduced. This is followed by solutions for several heterogeneous object modelling problems using distance fields and smooth approximate distance fields. First, interior distances and Voronoi diagrams are used to provide a more intuitive and predictable interpolation of attributes between features within shapes than the transfinite interpolation in Biswas *et al.* (2004). Secondly, the interpolation of volumetric properties through time for the shape transformation are devised by extending the transfinite interpolation to space-time. The last section shows how heterogeneous volumetric modelling operations can find applications in traditional shape modelling.

In chapter 4, several applications are described to illustrate the usability and usefulness of the various techniques and methods introduced in this thesis. First, several applications of convolution filtering are developed, including the shape preserving, smooth distance fields. Shape conforming volumetric interpolation is supported with colour interpolation and parametric design of volumetric microstructures within a shape. Space time transfinite interpolation is showcased afterwards using shape transformation between heterogeneous objects and microstructure control. Finally, morphological shape generation is shown with a specific chair generation problem.

In chapter 5, a summary of the problems and solutions presented in this thesis is given. The chapter concludes by outlining the contributions and a non-exhaustive list of potential future research.

Chapter 2

Background

In the introduction, we have highlighted the necessity of heterogeneous object modelling and some of the current challenges. In this chapter, we provide a quick overview of the different geometric (or shape) representations and their limitations. The strengths and weaknesses of each representation will help us understand how they carry over in heterogeneous object modelling.

We follow on with a survey on heterogeneous object modelling techniques which will show that scalar fields, and in particular distance field functions, are extremely useful for defining varying material and other physical properties of an object. Distance fields also provide a flexible solution to represent legacy objects with scalar fields.

Since heterogeneous volumetric modelling relies on distance fields, an in depth survey of distance fields is presented. We use this survey to show how such fields can be evaluated, but also show their more common uses for various applications.

The objectives of this work will be identified in the conclusion to this chapter. The problems tackled in this thesis will also be stated.

2.1 Object representation

In computer graphics, objects can be represented by their boundaries or their volume. This leads to two major classes of representation. Here, we provide a brief overview of each representation and its strength and weakness. While this section only deals with geometry, it will provide the basic knowledge required for the discussion of the state of the art in heterogeneous object modelling.

2.1.1 Boundary representation

Amongst Boundary Representations (BReps), parametric surfaces and polygonal meshes are the most popular. Both define a solid object by connecting several primitive elements of its boundary together. The object is therefore defined by its boundary (surface).

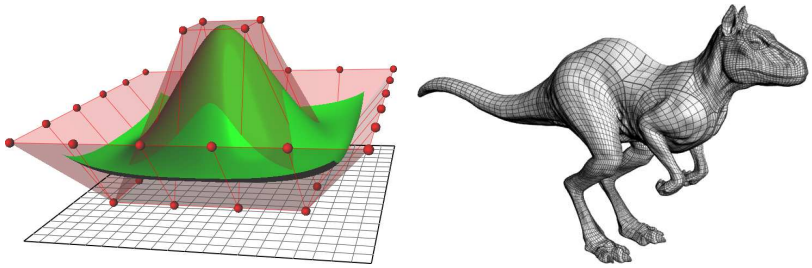
2.1.1.1 Parametric surfaces

The parametric representation maps from parametric space to another space, usually 3D Euclidean space. The parametric space is one dimensional for curves, two dimensional for surfaces, and three dimensional for volumes. For example, a curve in two dimensions can be defined as follows:

$$C(u) = (x(u), y(u)) \tag{2.1}$$

Here, u is the parameter, and $(x(u), y(u))$ is a point on the curve in Euclidean space. There are numerous parametric curve models used in computer graphics. Most of them use a set of control points, which define a path to interpolate or approximate [1]. The curve itself does not necessarily go through the control points (in curves such as B-Splines and Bézier curves). Amongst the many models, the most popular ones are

[1] In this case, interpolating means the curve passes through the control points, while an approximating curve will not necessarily pass through these points



(a) (b)

Figure 2.1: *(a) A NURBS patch (green) with its control net (red) and (b) The killeroo model is made of several NURBS patches. Killeroo model courtesy of Headus Pty Ltd.*

Bézier and Non-Uniform Rational B-Spline (NURBS) (Piegl and Tiller 1997). The latter one is a generalization of the former.

A NURBS surface can be constructed from a set of $n \times m$ control points. The NURBS surface is a tensor product of two NURBS curves.

NURBS curves and surfaces have many useful properties and the reader should refer to Piegl and Tiller (1997) and Farin (2002) for a more complete introduction. However, several algorithms rely on a few fundamental properties:

- The convex hull of the control points contains all the points on the curve or surface.
- A NURBS curve or surface can be split into several rational Bézier surfaces of the same degree.
- Multiple knots reduce the surface geometric continuity at that particular knot. The continuity at a knot is $n - k$ where k is the multiplicity of the knot and n the degree of the curve.

In order to create complex objects, several parametric surfaces (see Figure 2.1a) are stitched together (see example in Figure 2.1b). They are called parametric surface bounded solids. In order to define a valid solid, the following is required:

- The surface is an orientable manifold. Orientable manifold surfaces allow for T-junctions.

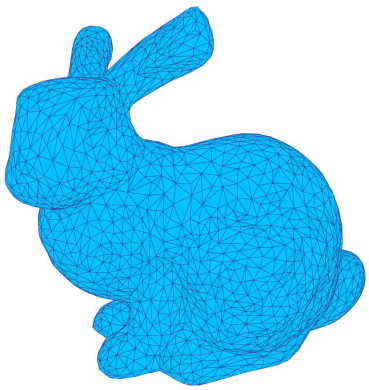
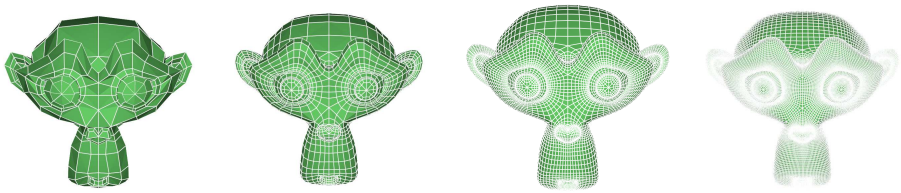


Figure 2.2: *The Stanford bunny model as a mesh*

- The surface excludes open boundaries.

Unfortunately, parametric surfaces are difficult to operate on. Numerous powerful operations are tedious or unstable, such as Boolean operations (Biermann *et al.* 2001), blending Boolean operations and offsets (Piegl and Tiller 1999). Parametric surface bounded solids are also subject to many common issues with boundary representations. Self-intersections are particularly problematic as it is difficult to interactively indicate self-intersections. T-junctions, while valid, are also a source of issues. When the boundary curve C_a of a surface is in contact with two boundary curves $C_{b,1}$ and $C_{b,2}$ equivalent to C_a, a T-junction is created. T-Junctions are often problematic because numerical inaccuracies can cause gaps and tessellation of the surfaces will lead to gaps and self-intersections. Fixing such issues often involves manual work, and it is still an open problem in research (Rajab *et al.* 2012; Pekerman *et al.* 2008; Krishnamurthy *et al.* 2008). Rendering parametric surface bounded solids often requires an intermediate step of conversion to a polygonal mesh, which can only approximate the model and can lead to additional defects, including self-intersections even if the parametric surfaces do not intersect.



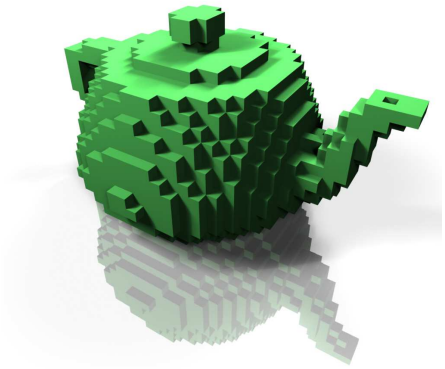
(a) (b) (c) (d)

Figure 2.3: *The Suzanne model (a) is subdivided iteratively (b,c,d). Suzanne model courtesy of Blender Foundation*

2.1.1.2 Polygonal mesh

A polygonal mesh is made of a set of vertices, a set of edges and a set of faces (see figure 2.2). An edge connects two vertices together, and n edges form a polygon. A triangle consists of three edges, and a quadrilateral (quad) consists of four edges. Meshes are widely used in animation, visual effects and video games. They have proven to be popular for numerous reasons. They can be rendered efficiently, they are easy to manipulate or animate and they all share a simple description (a list of vertices and a list of triangles). However, they often require large amounts of memory, and fail to produce smooth surfaces. To approximate smooth surfaces, meshes require a considerable number of polygons. Furthermore, similarly to NURBS bounded solids, polygonal meshes need to be manifolds.

Progressive meshes (Hoppe 1996) were introduced to adaptively change the number of polygons according to the needs. However, it requires a fine mesh first, and then creates coarser meshes. On the contrary, to get smooth objects from coarse meshes, subdivision surfaces have been used (Catmull and Clark 1998; Loop 1987; DeRose *et al.* 1998). Subdivision surfaces are a bridge between parametric surfaces and meshes. The mesh is used as a control grid, and faces are subdivided iteratively, as many times as needed (see figure 2.3). Vertices are moved according to a particular function which mimics parametric surface subdivision. Recently, T-Splines (Sederberg *et al.* 2003) have been introduced and permit T-Junctions and less restrictive modelling rules while maintaining a smooth surface.



(a)

Figure 2.4: *The Utah teapot model represented with voxels*

Polygonal meshes are popular because they allow the designer or artist to directly manipulate the surface. Nevertheless, polygonal meshes and their numerous extensions often lead to non-manifold objects. Self-intersections, holes and inconsistencies cause ambiguities in certain locations.

2.1.2 Volumetric representation

Boundary representations are convenient for designers who want to work with the surface of the object rather than with its volume. Because of the partial definition of the object, it often leads to problems such as holes, self-intersections and inconsistencies. While a lot of research works investigated fixing individual issues with BReps, the overall problem with BReps will always remain because a BRep does not describe a volumetric object but rather a set of boundary surface sheets. Volumetric representations have also been explored and proved to be more robust, but come with a different set of issues.

2.1.2.1 Voxels

The simplest volumetric representation is a discrete set of cells or voxels (short for volume elements) representing the list of cells occupied by

the volume. Typically, a voxel set is stored in a 3D array where each cell is set to *"full"* (inside) or *"empty"* (outside). Sometimes, voxels can contain more information such as density, colours or scalars. This is often the case of data acquired through MRI or CT scans. The simplicity of this representation makes it convenient for most geometric operations as described in Payne and Toga (1992); Cohen-Or *et al.* (1998). It is very robust, and volume rendering is a well investigated topic (Coat 2010).

However, this representation is resolution dependent, and as a consequence, often memory consuming. To capture medium sized details, the resolution has to be relatively high. A grid of 512 cells in each dimensions yields 2^{27} cells (more than 100 million cells). Additionally, if only point membership in the Boolean form is stored, the geometry cannot be smoothed and has a distinct boxlike aesthetic (see figure 2.4). Some attempts have been made to improve voxel visual appearance, but these techniques remove sharp features (Barthe *et al.* 2002).

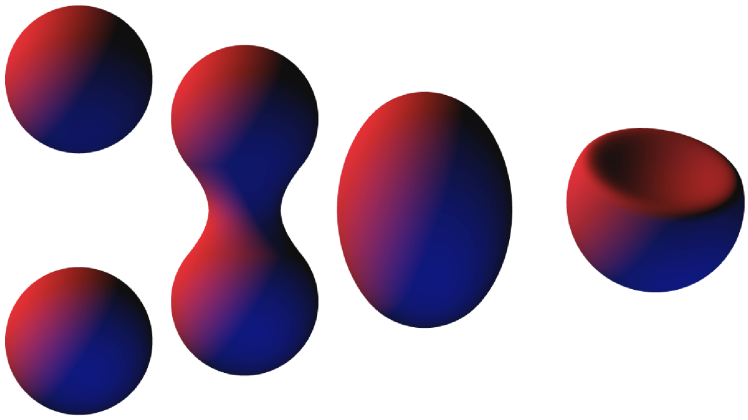
2.1.2.2 Implicit surfaces

An implicit surface is a conventional term to designate a zero-level (or another iso-level) point set defined by a continuous function of point coordinates in three dimensional space. A volume object bounded by an implicit surface is defined as follows:

$$\begin{cases} f(\mathbf{p}) < t & \mathbf{p} \text{ is inside the object} \\ f(\mathbf{p}) = t & \mathbf{p} \text{ is on the surface} \\ f(\mathbf{p}) > t & \mathbf{p} \text{ is outside the object} \end{cases} \tag{2.2}$$

where t is the iso-level, $\mathbf{p}$ is any point in space. The implicit surface is not explicitly described, but instead the function f provides information about the points belonging to the iso-surface with the given function iso-level.

In Blinn (1982), the author introduces the summation of Gaussian functions, which can be used to model electron density maps of molecu-



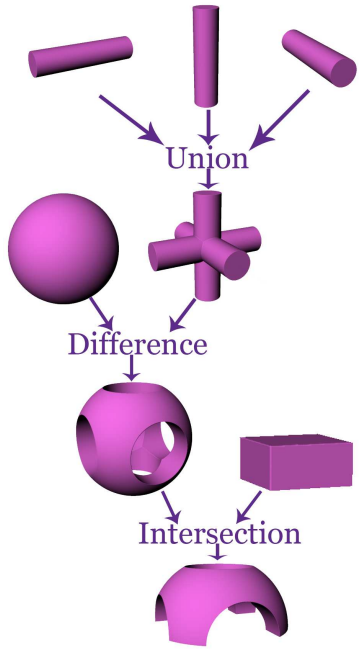
(a)

Figure 2.5: *Two metaballs blended together. Image from Wikipedia*

lar structures and artistic shapes (see Figure 2.5). Wyvill *et al.* (1986) extended the idea to introduce soft objects, which allow for local support and give smoother results. In Bloomenthal and Shoemake (1991), a set of simple primitives such as points, lines, rectangles and triangles are enveloped by using a simple bell curve function of the distance function to the primitives. These techniques are often called skeletal implicit surface primitives. The involved scalar field is called a compactly supported field (bump function), because the function takes the predefined value (iso-level) everywhere further away than a certain distance from the surface. Several other techniques are detailed in Wyvill *et al.* (1999). Metamorphosis is also a quite simple task with implicit surfaces (Galin *et al.* 2000). More recently, implicit surfaces were used to support a powerful sketch-based modelling tool in Schmidt *et al.* (2006). Their focus is to provide an accessible interface to novice users.

2.1.2.3 Constructive Solid Geometry

Constructive Solid Geometry (CSG) is related to volume modelling as it represents an entire object with its interior and surface points. CSG can be used to build complex solid objects from simple primitives and using simple operations. Primitives typically include the platonic solids, revolutions and simple sweeps. The object is defined by a tree of prim-

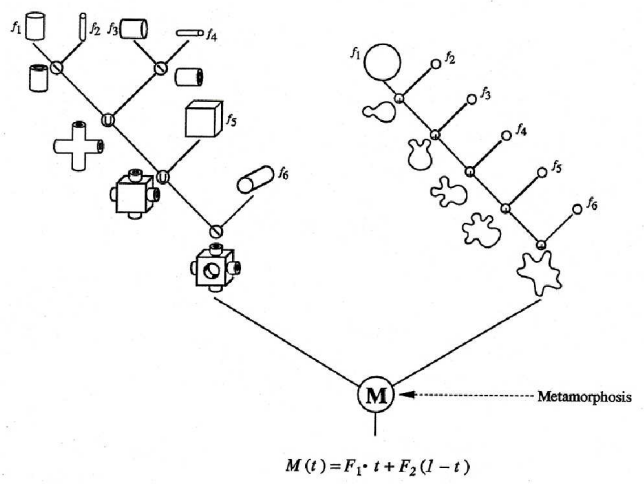


(a)

Figure 2.6: *A CSG-tree*

itives and operations. Each primitive defines a half space based on a predicate (a point is either inside or outside of the solid). Typically, operations are set regularized set-theoretic operations (union, intersection, difference) and affine transformations, although a few others are possible (e.g., twist, taper). An example of a CSG tree is shown in figure 2.6. CSG is is restricted to homogeneous solids defined by the binary (or ternary) point membership classification. A detailed study of the CSG basics can be found in Requicha (1980).

This representation is user-friendly. Any part of the tree can be modified at any point of the modelling process which is important for the design process as some decisions can be postponed. Additionally, CSG models are unambiguous and are always valid manifold models. The major issues with CSG are that CSG objects cannot be rendered directly by means of standard graphics hardware, and the set of operations is limited.



(a)

Figure 2.7: *An FRep model, combining mechanical parts with sharp features and organic looking parts. Image courtesy of A. Pasko, (Pasko and Adzhiev 2004)*

2.1.2.4 Function representation

FRep (Pasko *et al.* 1995) is a generalization of implicit surfaces and CSG. It defines an object with a procedurally evaluated function. The function evaluation is done through traversal of a tree data structure with operations as nodes and primitives as leaves. Operations can take any number of arguments allowing us to mix several models of different dimensionality into a single object. Figure 2.7 shows an example of an FRep tree, which incorporates sharp features and organic looking parts. FRep is capable of incorporating several other representations such as voxels, meshes and point clouds as primitives into the tree (Adzhiev *et al.* 2000). It also offers numerous advanced operations such as space time blending (Pasko *et al.* 2004), bounded blending (Pasko *et al.* 2005) and multi-scale cellular structures (Fryazinov *et al.* 2013). FReps are a powerful modelling representation, because they allow us to parametrise a range of models where the parameters of the operations can be adjusted to create numerous diverse objects from a single template. The tree also allows for modifying both operations and primitives at any time. This also makes FRep objects compact and resolution independent. The FRep was extended in Pasko *et al.* (2001) specifically for heterogeneous

volumetric modelling. It will be discussed further in the next section (section 2.2.5.1).

2.1.3 Discussion on object representations

Boundary representations are currently dominating and have clear advantages over volumetric representations. They are efficient to render, many users are trained to use surfaces, and they offer exact control of the surface. However, even some simple concepts such as Boolean operations are not trivial to implement, and most software packages allow for defects and errors to creep in, further damaging the object integrity.

Any alternative representation needs to be capable of import existing boundary representations (Adzhiev *et al.* 2000). The Function representation seems particularly suited for heterogeneous object modelling since it is able to represent shapes exactly, and to mix features of scales of different orders of magnitude. The FRep is also robust and guarantees that the object will be manifold, unlike BRep. However, with FRep it can be difficult to control the surface precisely.

2.2 Heterogeneous objects modelling

A heterogeneous object is an object which has varying materials and other physical properties across its volume. A material could abruptly change (*composite* materials) or gradually change from one end to another (*gradient* or *functionally graded* materials). While heterogeneous volumetric modelling often refers to materials, it is not limited to materials. Heterogeneous object modelling could be used to describe arbitrary attributes which could feed back in the shape modelling process.

The object representations for heterogeneous object modelling are classified in three major classes by Kou and Tan (2007): *composite* models, *evaluated* models and *unevaluated* models. Composite models are made by combining several sub-objects and are made of both evaluated

and unevaluated models. Evaluated models have a finite number of elements providing information about the material properties in a point or volume in space. *Evaluated* models are represented using voxels or volume meshes (e.g. tetrahedral meshes).

The other representations, the *unevaluated* models, represent volumetric properties through an exact representation such as a continuous scalar field. In Kou and Tan (2007), scalar fields are split into various sub categories (explicit function, implicit function, control feature based methods). In this section, we review the different representations for heterogeneous object modelling and the methods available. A more exhaustive review is presented in Kou and Tan (2007). The unevaluated models are represented using control point based models (such as B-Spline volumes) or scalar fields.

Composite models can combine evaluated and unevaluated models. In Kumar *et al.* (1999) and Wang and Wang (2005), an object is split into regions, and each region has a single *"material class"*. A material class can be evaluated or unevaluated.

2.2.1 Surface based heterogeneous object modelling

A single BRep object can only represent an object made of a single material (homogeneous object) since only its boundary is defined. Composite objects can be made by combining several parts to make a single object. Some techniques have been used to alter the surface material attributes through the use of textures but it is limited to the parametrization of the object surface and does not extend easily to the rest of the volume. Therefore, texturing an object cannot be considered true heterogeneous object modelling technique since it does not precisely define attribute values within the object volume.

As mentioned above, texture based methods are capable of creating the illusion of heterogeneous objects. However, for a texture to be applied, a 2D parameterization of the surface has to be done, which can be arduous (Lévy *et al.* 2002; Yoshizawa *et al.* 2004). The 2D parame-

terization can be bypassed by the use of superposed patches (Pedersen 1995) and decals (de Groot *et al.* 2014), but once again, this technique focuses only on surface information. The extension of the surface information is also problematic. It can be done through Mean Value Coordinates (MVC) (Ju *et al.* 2005) or Green coordinates (Lipman *et al.* 2008). However, such solutions tend to extrapolate the information in an unintuitive manner, and they do not provide user-directed control away from the surface.

In Kumar and Dutta (1997), a framework to support heterogeneous object modelling using boundary representation was introduced. The model can only use a set of pre-defined materials (material palette). The model becomes an aggregation of homogeneous solid components. The Boolean operations are extended to operate on the material components as well as the geometry. The material palette gives the illusion of gradual change, but it is only able to represent a set of a number of layers. Furthermore, maintaining such a complex set of objects as more operations are applied to the geometry and its attributes makes this method unsuitable for complex heterogeneous objects.

2.2.2 Voxel based heterogeneous object modelling

Extending voxels to support heterogeneous object modelling is a straightforward task. Voxel lattices have been used to describe object internal attributes as for instance in Nielson (1993). According to Chen and Tucker (2000), the user can easily control the values of the attributes through operations which are applied simultaneously to object geometry and its attributes. The software tool Michalatos and Payne (2015) relies on voxels for both geometry and attributes. The users can control the geometry and the object attributes through simple operations. Many of these operations were inspired by image processing tools such as paint brushes. Voxels are also a natural choice for medical applications, since MRI and CT scans provide voxel data. MRI scans for instance, provide information about magnetic properties.

Voxels are popular because they are simple and efficient. This model can represent any number of attributes (limited by memory only), and is trivial to query. Additionally, it is straightforward to implement. Therefore, voxels can provide smooth attribute changes. They are particularly suitable for physical simulation. In Cho and Ha (2002), functionally graded materials were used to model the volume fraction attribute to optimize the heat resistance. The simulation is performed on a regular lattice in 2D (pixels), and could be extended to 3D using voxels. In Zhang *et al.* (2004), voxels are used in conjunction with numerical simulation to define the optimal material properties such as material microstructures and material constituent compositions. This simulation based method leaves little room for user input and design. In Wu *et al.* (2005), the user can define features (such as faces and edges) and use those features to make gradual change in material attributes across the voxel grid. However, the gradual change is performed on a relatively small grid, while the geometry is represented by the mesh. The mesh is kept for the slice accuracy, and the voxel are used to provide colour information. Voxels are also gaining popularity in 3D printing since it can easily be sliced or control the 3D printing process. A few printing services and 3D printers are starting to accept voxels such as Shapeways (Shapeways 2015).

However, the choice of the resolution (the size of a single voxel) can be problematic. If the size of a voxel is too small, the memory required to store such a grid might be too large. This also means a need for a huge processing power. Wang *et al.* (2011) allows the user to create, edit and visualize in real-time a complex heterogeneous object made of voxel lattices of different scales at the cost of memory. In 3D printing, for instance, Solidscape 3D printers print at a 0.00625 mm resolution, on a 5 cm wide bed. To represent this print bed, 512.10^9 voxels would be required. If the size of a voxel is too large, then accuracy is lost and the voxels cannot represent high frequency attributes or thin geometric features. In practice, it means large objects with small features or high frequency attributes cannot be accurately represented. Crassin *et al.* (2009) has introduced a method to reduce the memory footprint of voxels

by storing a sparse octree of smaller voxel grids, but it only captures the region around the surface, and therefore cannot store information except close to the surface.

Voxels present a simple approach to heterogeneous volumetric modelling but are found lacking in several areas. First, by its nature, a voxel representation cannot define a continuous and exact gradual change of an attribute. It also leads to approximations which accumulate and deteriorate as additional operations are performed. Secondly, the choice of the resolution limits the difference in scale of the features of both the object and of their material properties. Memory issues also rapidly arise.

2.2.3 Volumetric meshes

Volumetric meshes are polygonal meshes which can represent the interior of the object. Volumetric meshes are commonly made of tetrahedrons. There are numerous methods to generate tetrahedral meshes from polygonal mesh surfaces (Shewchuk 1998; Si and TetGen 2006). Volumetric meshes can be uniform or adaptive. An adaptive mesh can have denser areas to better approximate topological events or attribute variations. Volumetric meshes are often used for finite element analysis, which can be used to evaluate the robustness of an object and many other physical properties such as heat resistance. Commercial software tools such as Autodesk (2015) often rely on volumetric meshes to perform physical and mechanical analysis of the model. Xu *et al.* (2015) relied on volumetric meshes to design and optimize the internal elasticity of an object to conform to the user defined constraints. Schumacher *et al.* (2015) further explored the idea by blending various microstructures to match the target elasticity, allowing users to print deformable objects with a single material.

Jackson *et al.* (1998) used volumetric meshes to define functionally graded materials, where some user-defined parts of the object are assigned values, and the nodes of the mesh are used to interpolate and blend the known values. The mesh can be tessellated at a higher resolu-

tion where needed. Representing heterogeneous objects with volumetric meshes also has the benefit of allowing local edits by the user.

However, tetrahedral meshes have several major issues. First, the memory requirements can be prohibitive. Furthermore, it is only an approximation of both the material distribution and the geometry. The attribute can only be interpolated inside the tetrahedrons, and the quality of the interpolation depends on the meshing density. The points on the surface of the object may not have a material distribution defined if a volumetric mesh does not match the surface exactly. Finally, a change in the material distribution functions, or in the geometry of the object will require a new volumetric mesh generation, which is a time consuming process. Adzhiev *et al.* (2002) used both cellular structures and scalar fields to model heterogeneous objects. However, in order to move between representations, a conversion needs to happen, which can be time consuming and often results in a loss of accuracy.

2.2.4 Parametric volumes

Parametric volumes can be used to represent arbitrary attributes with true gradient materials, which can be controlled accurately. Martin and Cohen (2001) used tri-variate NURBS models to represent a number of attributes such as refraction and density. This method makes it easy to define exact values at user defined points, and provides controllable smooth gradual changes of the attributes. However, it requires the model to have a 3D parametrization of the object space.

In Qian and Dutta (2003b) and Hua *et al.* (2004), the geometry is represented with a B-Spline model, and the volumetric attributes are controlled at the surface. A B-Spline volume is fitted to the object, and the interpolation is performed through diffusion. These method lack control of the volumetric attributes, and they rely on the fitting algorithm, which is not trivial to implement for arbitrary objects. However, Qian and Dutta (2003b) can represent time-dependent heterogeneous objects, where attributes are functions of time. In this case, the

initial state is defined by the user. The diffusion process will then converge towards a stable state. An example of this method applied to a real-world problem for time dependent design of heterogeneous object modelling was given in Qian and Dutta (2003a).

Samanta and Koc (2005) use B-Splines to interpolate geometry and attributes. This partially helps with shape awareness, as the interpolation seems to follow the shape and is directly linked with the geometry. However, the entire process relies on a discretization step, and is also limited in complexity. The material features are automatically detected to best respond to the physical requirements but user input is very limited afterwards.

2.2.5 Scalar fields based methods

Numerous heterogeneous structures can be approximated using procedural solid texturing (Perlin 1985). Solid texturing has been employed to reproduce complex natural materials such as wood, marble and rust (Ebert 2003). Most effects are achieved by layering different smooth noise functions with varying amplitudes and frequencies. The produced scalar field is then passed to a transfer function which converts the function value into a colour or other material attribute. Worley (1996) introduced a cellular noise function which allows the creation of skin or pebble rocks. Solid texturing is resolution independent and cheap both in memory and computation time. Its main drawback is in user control and intuitiveness. It is particularly challenging to reproduce an exact pattern using only solid texturing and it requires an advanced knowledge to reproduce the composition of a specific object using solid texturing. This process was simplified in Kopf *et al.* (2007), where a solid texture function was created from 2D images. However, the resulting solid texture is computed and stored in a discrete volume and involves an expensive optimization step. Nonetheless, it remains state-of-the-art in solid texturing.

HyperFun (Adzhiev *et al.* 1999) is a language which enables advanced users to create geometric models, including heterogeneous object models.

The user can create an object using an explicit function of point coordinates. However, modelling precise and complex models within HyperFun requires skilled users who can program and provide the function for the material distribution.

All the above methods are not user-friendly. Constructive methods of material distributions and features based methods presented below have been used to provide more user-friendly approaches. These methods are not mutually exclusive.

2.2.5.1 Constructive methods

There is a clear need for more exact control from the user. In Kou and Tan (2005), boundary representations are used for geometry, but the authors introduce a Heterogeneous Feature Tree for material distribution which is based on scalar fields and constructive geometry. For any point in space, the material is a blend of all the material features of the tree. The features are defined during the modelling process by the user.

A more general formulation was introduced in Schmitt *et al.* (2001) and Pasko *et al.* (2001), and enabled FRep to support heterogeneous object modelling for both composite and gradient materials. Constructive Hypervolume modelling allows the user to model geometry along with any other attribute. The object is defined by its geometric set and a set of its pointwise attributes. The tree of each attribute, and the geometric tree can be different and do not need to be similar. Formally, a Constructive Hypervolume model was defined in Pasko *et al.* (2001) as follows:

$$o = (G, A_1, \ldots, A_i) : (F(\mathbf{p}), S_1(\mathbf{p}), \ldots, S_i(\mathbf{p})) \tag{2.3}$$

where,

- $\mathbf{p}$ is a point in Euclidean space
- F is a real-valued function of point coordinates which represents the geometry.

(a)

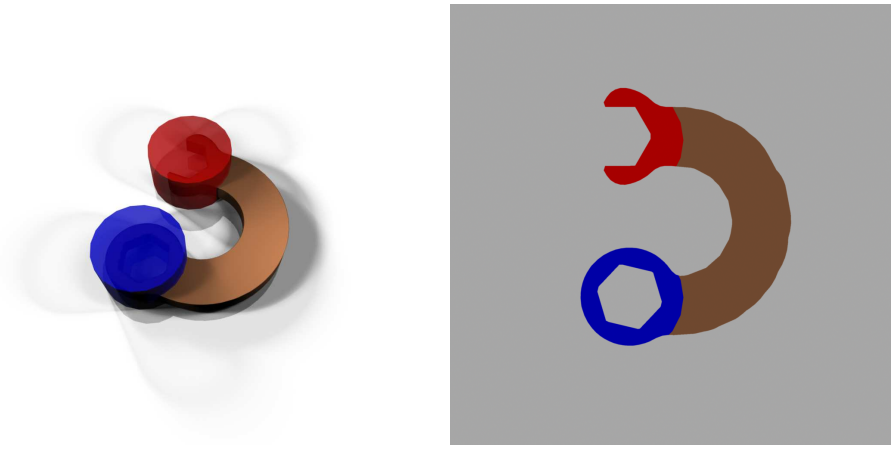
Figure 2.8: *A spring and a gradient material using sweeping. Image from Shin and Dutta (2001)*

- S_i is a function of point coordinates representing an attribute.
- o is the object
- G is the geometry of the object
- A_i is an attribute of the object

It is important to note that while F requires at least C^0-continuity, S_i can be discontinuous, since attributes could have discontinuities. This solution is user-friendly because the object is built from simple primitives and simple operations. Additionally, changes anywhere in the tree can be made effortlessly at a later stage of the modelling process.

Shin and Dutta (2001) proposed a similar solution, using CSG as an inspiration, and introduced new primitives and operations to construct a tree for both geometry and material attributes. Blending of materials is also supported. This method, and the other constructive methods allow to build complex heterogeneous object where the material transitions are performed with respect to the shape. The figure 2.8 shows a disk swept along a curve. The material transition is based on the sweep curve.

Kou and Tan (2005), Pasko *et al.* (2001) and Shin and Dutta (2001) enable the user to edit the geometry alongside the material attributes. Both allow to create geometry dependent material distributions by build-



(a) (b)

Figure 2.9: *Feature based modelling and material gaps. (a) shows how the features are set up, (b) shows a slice of the model, with the features in blue and red, and the material gap in brown*

ing trees or functions which are adapted to the shape.

2.2.5.2 Feature-based heterogeneous modelling

An important principle of heterogeneous volumetric modelling is *feature*-based heterogeneous modelling where geometric features are defined within the volumes. Such features are often user-defined, and can be polylines, curves, points, surfaces or even solids. These features have prescribed material properties, and are called *material features*. The point set of all the points which do not belong to any material feature is called the *material gap*. Feature based methods are used to find values in these material gaps. Figure 2.9 shows the material gap (in brown) given two features (in blue and red).

Several authors introduced methods where colours or other attributes would fall off depending on the distance to a feature such as an edge or a line segment. Liu (2000) uses the distance to the boundary of the object to gradually change colours between the surface of the object and its interior. In Zhou *et al.* (2004), any surface, point or line can be used as a feature, and a function of the distance is associated with a feature. In Khoda *et al.* (2013), the medial axis of the shape is extracted, and

then a gradient porosity from the medial axis (dense) to the boundary of the object (sparse) is used to automatically control the following process to generate a structure with adaptive porosity. In Liu *et al.* (2004), two materials can be interpolated by letting the user define a feature (such as a solid), and the interpolation is performed within the distance to this feature. Kumar *et al.* (1998) also used distances to interpolate between two materials. Some geometric features represent a material, and the interpolation is based on a function of the ratio between the two distances. Bhashyam *et al.* (2000) provides a list of functions to interpolate between various materials to optimize for some desired properties. Park *et al.* (2001) introduced Volumetric Multi-Texturing (VMT) to define gradient materials. Some features are defined using implicit surfaces such as algebraic surfaces, and then the values of the function are mapped onto material or density values.

In Siu and Tan (2002b), the features controlling the gradient materials are called grading sources. Siu and Tan (2002b) use any number of arbitrary features, of any type or dimensionality such as line segments, planes and points. Each feature provides a value for each modelled attribute, and the material distribution is expressed in function of the distance to those grading sources. The control of the graded materials is made easier through these simple control features, and Siu and Tan (2002b) extends the usual operators (union, intersection and difference) with new capabilities for heterogeneous object modelling. The control of the gradual change is user-friendly. Moreover, this solution can easily be integrated into existing applications. However, it does not provide a global method, since the sources have a limited influence in space. Wu *et al.* (2005) provided a similar structure, where the material composition is successively built by providing material features with their material values and distances to control the length of the fall-off.

In Rvachev *et al.* (2001), transfinite interpolation was used to interpolate material properties given any number of features. The problem of transfinite interpolation is to construct a function which *"takes prescribed values and/or derivatives on some collection of point sets"* (Rvachev *et al.* 2001). The point sets with prescribed values can be any

point set, such as points, lines, surfaces or spatial regions, each represented by a real valued function. The interpolation is performed by using scalar fields generated from each material feature. The scalar fields can be distance fields (a minimal Euclidean distance from any point in space to the object boundary) as in Biswas *et al.* (2004), or scalar fields built with R-Functions (Rvachev 1982) as in Rvachev *et al.* (2001). The R-Functions are chosen to have C^1 continuity, however, the R-Functions need to be carefully selected to preserve the distance properties. In fact, the authors used smooth R-functions to build *"smooth distance-like functions"*. Both Rvachev *et al.* (2001) and Biswas *et al.* (2004) argue that distance approximation is necessary, and smoothness is desirable. As stated in Biswas *et al.* (2004), the distance provides predictability, and the smoothness provides smoothness to the attribute interpolation. Hongmei *et al.* (2009) proposed a similar method which provides more control over the rate of change of the attributes.

There are several interpolation schemes, however, inverse distance weighting formulated in Rvachev *et al.* (2001) (eq. 2.4) is the most popular (Kou and Tan 2007).

$$w_i(\mathbf{p}) = \frac{\prod_{j=1;j\neq i}^{n} d_j(\mathbf{p})}{\sum_{j=1}^{n} \prod_{k=1;k\neq j}^{n} d_k(\mathbf{p})} \tag{2.4}$$

where

- $\mathbf{p}$ a point in space
- $d_i(\mathbf{p})$ is the unsigned distance of the point $\mathbf{p}$ to the i-th feature
- n is the number of features
- $w_i(\mathbf{p})$ is the weighting of the feature for the point $\mathbf{p}$

If only two features are used, this simplifies to:

$$w_a(\mathbf{p}) = \frac{d_b(\mathbf{p})}{d_a(\mathbf{p}) + d_b(\mathbf{p})}$$
$$w_b(\mathbf{p}) = \frac{d_a(\mathbf{p})}{d_a(\mathbf{p}) + d_b(\mathbf{p})} \tag{2.5}$$

The transfinite interpolation was applied in a similar fashion in Fryazinov *et al.* (2013) in order to interpolate between parameters of two given volumetric microstructures. This is a good example of the link between shape modelling and the modelling process related to an arbitrary volumetric attribute. Since distance properties are important to have intuitive interpolations, Fayolle *et al.* (2006) provides FRep C^1 continuous operations and primitives which approximate the distance. The continuity is necessary to avoid *"undesirable singularities in the material distribution, like stress or concentrations"* (Fayolle *et al.* 2006).

2.2.6 Discussion

As Kou and Tan (2007) stated in their survey of the different representations, evaluated models are not exact nor compact which can be problematic. For instance, 3D printers are increasingly more and more precise. The Connex 3D printers by Stratasys can print at 16-micron for objects up to 25 centimetre wide. Evaluated models cannot simultaneously be compact, accurate and computationally efficient. Unevaluated models are resolution independent, and scalar field based methods are capable of representing complex material distributions at various scales without loss of accuracy, and remains compact.

In the unevaluated representations, two scalar field approaches stand out: constructive methods and feature-based methods, both of which often rely on distances. As our survey shows, the use distance fields and feature based modelling for representing gradient material and other volumetric attributes is popular.

Feature-based methods provide a solution to easily set up features and

let the users control the material distribution everywhere. Additionally, feature based methods are easily integrated into existing CAD systems. However, these methods rely upon functions of the distance to the features, never taking into account the shape of the object being modelled. Additionally, using exact distance functions can cause abrupt changes, but building smooth functions is not always straightforward for arbitrary features.

The constructive methods can build a complex heterogeneous object where the gradient materials are 'following' the shape. Additionally, several authors have shown that constructive methods can build complex heterogeneous objects alongside the object geometry. However, it requires the user to build the functions carefully and cannot be easily performed at a later stage of the modelling process. Additionally, the scalar fields produced by FRep trees do not generally approximate distance fields, preventing the use of distance-based methods with FRep trees.

2.3 Distance fields

In section 2.2, we mentioned that distances are a fundamental element of heterogeneous volumetric modelling. Previously, several authors (Siu and Tan 2002a; Biswas *et al.* 2004; Fayolle *et al.* 2006) have supported this claim.

We can distinguish continuous distance fields from discrete distance fields. The distance field is defined at any point in space and is a minimal Euclidean distance from the given point to the given object boundary. Distance fields can have different meanings throughout the literature. It can be exact or approximate, discrete or continuous, and is used with different sets such polylines, polygonal meshes, point clouds, NURBS bounded solids, and voxels. A more exhaustive survey of distance fields, discrete distance fields in particular, is presented in Jones *et al.* (2006). In this thesis, we will employ the term distance field to mean a continuous exact distance field by default.

Distance fields can serve a dual role in heterogeneous volumetric modelling. First, distance fields are a convenient way of representing various BRep objects or geometric features with scalar fields. Distance fields are also not limited to BRep objects and can also be created from simple geometric elements. Secondly, they can be used to define material *features*. It is desirable to have distances to material *features* or approximations of distances to avoid unexpected results in the attribute interpolations. The smoothness of the various scalar fields (distance fields or approximations) has to be considered, if some degree of smoothness of the interpolation is required. Biswas *et al.* (2004) used interpolation across the voxel grid to provide a smooth function, while Fayolle *et al.* (2006) built FRep objects with distance field primitives and smooth operations which approximate the distance. Biswas *et al.* (2004) however can only approximate the material features, and Fayolle *et al.* (2006) cannot directly use BRep to define material features.

In this section we will:

- define distance and signed distance fields,
- study how to efficiently evaluate the signed distance function,
- survey alternative solutions to represent BRep objects with scalar fields and their drawbacks,
- survey some of the many applications of distance fields.

2.3.1 Definition & terms

The *unsigned* distance to a set Σ is the distance from the given point $\mathbf{p}$ to the closest point $\mathbf{x}$ on Σ.

$$dist(\mathbf{p}; \Sigma) = \inf_{\mathbf{x} \in \Sigma} \|\mathbf{x} - \mathbf{p}\| \tag{2.6}$$

where inf defines the infimum operator, which returns the smallest element of a set. The distance to a set does not require a volume and can be calculated against a set of lines. However, when the set defines a

solid object $\mathbf{S}$, a *signed* distance can be more interesting for many applications. The sign of the distance is used to indicate whether the point $\mathbf{p}$ is inside or outside the solid object. We arbitrarily chose the sign to be positive inside, and negative outside.

The sign is defined by:

$$sign(\mathbf{p}; \mathbf{S}) = \begin{cases} +1 & \mathbf{p} \in \mathbf{S} \\ -1 & \mathbf{p} \notin \mathbf{S} \end{cases} \tag{2.7}$$

Therefore, the signed distance is written as follows:

$$f(\mathbf{p}) = sign(\mathbf{p}; \mathbf{S}) \cdot dist(\mathbf{p}; \Sigma) \tag{2.8}$$

Signed distance fields were first introduced in Rosenfeld and Pfaltz (1966) where the points were sampled along a regular grid. In Payne and Toga (1992), the definition is given first as a continuous function, although the implementation is also based on a regular grid.

Signed distance fields have many notable properties. The gradient is not defined everywhere, but where it is defined, the magnitude of the gradient is one. However the gradient can be undefined if there are two distinct points on the surface at the same distance of the query point $\mathbf{p}$. The point set of all the points which do not have a single closest point is called the medial axis.

2.3.1.1 Discrete distance fields

A discrete distance field stores the shortest distance to a set of primitives in a node of a voxel lattice. This naive approach was used in Payne and Toga (1992). In Green (2007), a black and white image is converted to a discrete signed distance. Each pixel provides the signed distance to the first pixel of the opposite value.

The main advantage of voxel grids is that they can be computed once

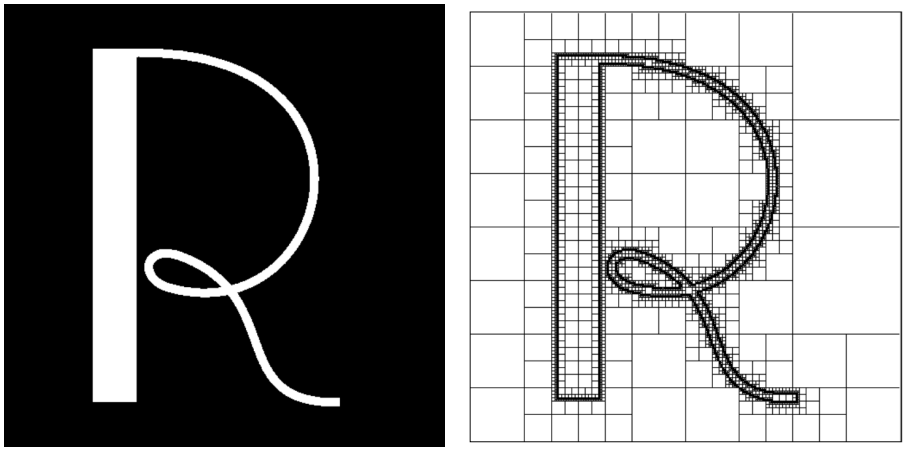
and then stored. If the distance needs to be evaluated in-between two voxels, then a trilinear interpolation can be used. Despite this interpolation, such a discrete field is considered exact if each voxel value stores the exact distance. The quality of the field is affected by the resolution of the grid, and certain features may be lost or incorrectly approximated.

Computing the distance on a regular grid can be vastly improved. In Strain (1999), a quad-tree was used on a 2D binary image to compute its corresponding distance field. Scanning methods Sigg *et al.* (2003); Nguyen (2007) are also popular for discrete grids. They use Voronoi diagrams where the elements of the set cast a prism onto the grid. Such an algorithm is easily parallelised and in particular suits graphics processing units (GPU). The work Erleben and Dohlmann (2008) particularly stands out because it can handle the distance field conversion for meshes with defects, which is a common issue with boundary representations.

2.3.1.2 Approximate distance fields

The computation of distance field values on the voxel lattice is a time consuming process at high resolution. Therefore, approximate solutions can improve the computation times at the cost of reduced accuracy. Most approaches to approximations of distance fields try to preserve good accuracy in a narrow band around the boundaries. The first pass fills the voxel cells, which intersect with the boundaries, the second pass then propagates the values using various distance transforms (DT) Rosenfeld and Pfaltz (1966).

Another alternative to reduce the amount of data is to use an octree instead of a regular lattice. In Frisken *et al.* (2000), the distance field is stored in an octree called Adaptively sampled Distance Field (ADF). The nodes of the octree are only further subdivided if the node intersects with the boundary of the object (see figure 2.10). The values in-between the nodes are then interpolated. This provides a high accuracy field close to the surface. However, the overall field is a crude approximation with poor properties. Discontinuities can happen at the borders of nodes of different resolution, and like any voxel lattice, some features can be lost.



(a) (b)

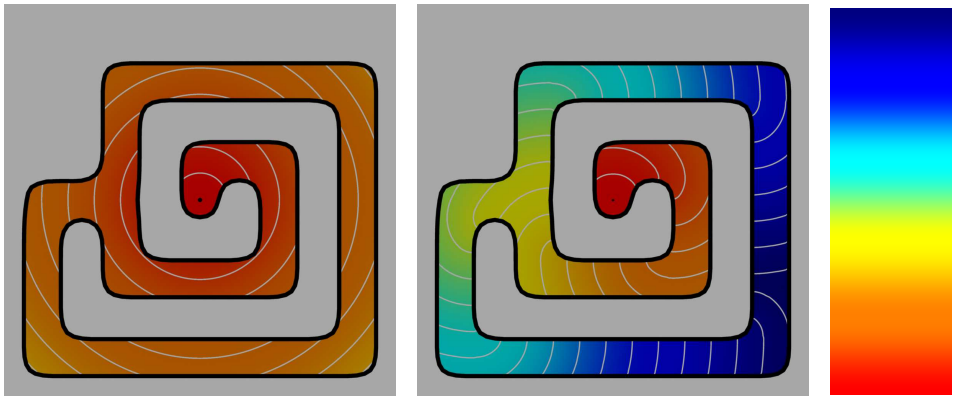
Figure 2.10: *The letter R as a binary image (a), and its Adaptively sampled Sistance Field ADF (b). Images from Frisken* et al. *(2000)*

2.3.1.3 Interior distances

The distance to the boundaries is useful for heterogeneous volumetric modelling, however, other classes of distances can also be beneficial. For instance, if a concave shape is used, and the transfinite interpolation (Biswas *et al.* 2004) is applied with the distance fields of the features, it is possible to have counter intuitive effects. One of the reasons is that Euclidean distances do not take into account the shape of the object when evaluating distances of each feature. This problem is well known in shape deformation, and has been identified in Levi and Levin (2014). Instead of Euclidean distances, interior distances are used.

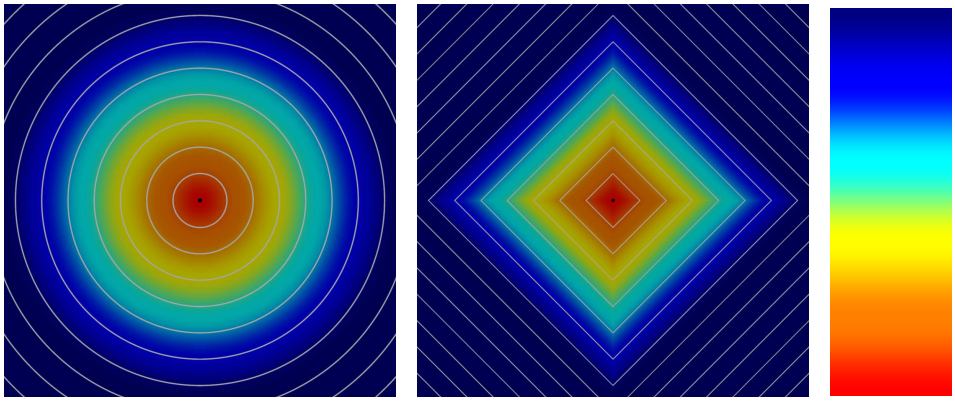
The interior distance is the length of the shortest path between two points within an object so that the path lies entirely inside the object. Figure 2.11 shows the distance field (left) and the interior distance field (right) side by side on a complex shape. Interior distances better reflect the perceived distances within a shape. For this reason, interior distances should also be considered for heterogeneous volumetric modelling.

The problem is closely related to the shortest path algorithms, which leads to the Djikstra's algorithm (Knuth 1977). The Dijkstra's algorithm can only work on a set of nodes however, and cannot be used continuously. A discrete grid along with the interpolation can provide



(a) (b) (c)

Figure 2.11: *The difference between interior distances (b) and Euclidean distances (a). Values range from zero (red) to one (dark blue), see (c).*



(a) (b) (c)

Figure 2.12: *Euclidean distance to the origin (a) and Manhattan distance to the origin (b), and the values to colours correspondence map (c)*

an approximation of the function, given that a point membership function exists. In three dimensions, the distance is propagated from the source feature unless it crosses the boundary. The typical trilinear interpolation used on a regular lattice will introduce numerous undesirable C^1 discontinuities. This can be avoided using other interpolation methods such as the one used in Freytag *et al.* (2011). However, the distance between two points on this lattice will lead to Manhattan distances. The Manhattan distance is the travel distance along lattice lines, which often leads to undesirable patterns (see figure 2.12b). It is possible to improve on these results by using more connections between the voxels (including diagonals) but this only reduces the problem.

The problem of finding interior distances within a shape is also known as the boundary value problem of the Eikonal equation in other fields such as computer vision. A numerical solution to this problem is the fast marching method introduced by Tsitsiklis (1995) for 2D domains and its generalization in Adalsteinsson and Sethian (1995). The fast marching method runs on a lattice which has its boundaries marked, and tracks the evolution of a curve in 2D and surface in 3D which expands from the source. The surface is never explicitly defined, but instead the lattice is filled with values following an advancing front. Recently, Hassouna and Farag (2007) introduced a method which is accurate and close to the analytical solution while maintaining the same complexity as Adalsteinsson and Sethian (1995).

The shortest path between two points with polyhedral obstacles was proven to be an NP-hard problem in Canny and Reif (1987) and remains a challenging problem. Some approximate solutions were provided in Papadimitriou (1985) and later in Choi *et al.* (1997). A continuous function for interior distances within mesh boundaries was introduced in Rustamov *et al.* (2009). The authors used barycentric coordinates to calculate pseudo interior distances. The shortest path from the point to all the vertices is first computed, using one of the previous methods, and then mean value coordinates (Ju *et al.* 2005) are used to extrapolate within the volume. Since this method uses MVC (Ju *et al.* 2005), it suffers from the same efficiency issues (linear time complexity per query),

but the field is smooth and is at least C^1 continuous. The linear time complexity is problematic since several millions samples will be required, sometimes on very large meshes (several million polygons). For instance, a 25 cm cube at a 600 dpi resolution would require approximately 6000^3 samples.

Another solution to finding approximate interior distances is based on the diffusion distance between two points. This can be achieved through heat diffusion (Coifman *et al.* 2005) or random walk distances (Yen *et al.* 2007). These solutions can be expensive and rely on a discretization of the volume. More recently, Crane *et al.* (2013) introduced a method based on heat diffusion to evaluate geodesic distances. This method seems to extend easily to arbitrary surface representations and this simple yet powerful principle should extend to interior distances equally using the boundary conditions. The heat diffusion method also allows for smoothing resulting in C^1 continuous fields, which is a desirable property.

Most of the work in heterogeneous object modelling focused on Euclidean distances. In most cases, Euclidean distances can be substituted by interior distances in order to provide more intuitive and predictable results for methods which rely on distances to features. Interior distances are used in this thesis in section 3.3 to improve on the method presented in Biswas *et al.* (2004). A simple method to calculate interior distances is also provided in appendix.

2.3.2 Representation of boundaries by scalar fields

Although distance fields can be calculated for many types of features and geometric representations, they were mostly studied in connection to meshes and non-uniform rational B-Spline (NURBS) bounded solids. We give a brief overview of the techniques used to query the distance to objects in those two representations.

2.3.2.1 Polygonal meshes

For polygonal meshes, many scalar field formulations have been investigated. We first overview the scalar field representations and assess their pros and cons, and then investigate more in depth exact distance fields.

Scalar field Methods for representing scalar fields of meshes can be distinguished as exact and approximate ones. In the exact methods, the iso-value is guaranteed with the finite machine precision to be zero only on the surface of the initial mesh, meaning that all the features of the initial models are preserved with the scalar field representation.

In the approximate methods, many techniques rely solely on the point cloud generated from the mesh to approximate the polygonal mesh. They often rely on a dense point cloud. Radial Basis Function (RBF) (Savchenko *et al.* 1995; Carr *et al.* 2001; Morse *et al.* 2001; Yngve and Turk 2002) use a weighted sum of basis functions. The choice of the basis function has a considerable impact on the shape and efficiency of the function evaluation. If the basis function has a compact support, then only a sparse linear system needs to be solved. Otherwise, the process will result in a non-sparse system, making it impractical for large point clouds. Most RBF-based methods produce smooth surfaces but are slow to evaluate.

In the Multi-level Partition of Unity (MPU) (Ohtake *et al.* 2003) or Moving Least Squares (MLS) (Turk and O'Brien 1999b), some given approximation error is allowed. In Turk and O'Brien (1999b), the constraints can be set so that it should interpolate the surface exactly, however, numerical instabilities make it impractical. All those approximate methods only approximate the surface, and the field approximates the distance poorly.

An exact field to a polygonal mesh can be obtained by representing the object with set-theoretic operations on the half-spaces bounded by planes passing through polygonal faces (Fryazinov *et al.* 2011) while keeping the distance property of the resulting function. These methods

provide continuous functions, but the distance query can be slow and numerically unstable especially for large input meshes. A smooth approximation of the distance field, called signed L_p distance fields, which can preserve the exact surface was introduced in Belyaev *et al.* (2012). This method is based on mean value coordinates (Ju *et al.* 2005). L_p distances have a linear complexity making them unsuitable for large meshes as unevaluated methods often require a large number of samples.

Minimum distance to a mesh evaluation The most naive solution of evaluating the distance to a mesh would be to iterate over every triangle of the mesh in order to find the shortest distance. The distance from a point to a triangle can be found through its barycentric coordinates (Ericson 2004, p. 136), vector calculus (Eberly 2008) or by simplifying the problem using affine transformations (Jones 1995). However, the performance can be greatly improved using various data structures such as octrees (Payne and Toga 1992) or bounding volume hierarchies (Larsen *et al.* 2000, 1999).

A bounding volume hierarchy is a structure which bounds the surface elements in a hierarchical manner. At the highest level (root), a bounding volume bounds the entire object. The root stores two children, both of which also bound the elements they contain. Each node in the tree bounds only a subset of the elements of the original object. The elements can be polygons for meshes, patches for parametric surfaces, or any element which can be bounded. Typical bounding volumes are axis aligned boxes (Havran 2000), oriented boxes (Gottschalk *et al.* 1996), or spheres (Quinlan 1994). In Larsen *et al.* (2000), the bounding volume is a rectangular swept sphere volume. This is more efficient because it provides tighter bounds for triangles than spheres or axis aligned boxes, and the distance to the rectangle remains simple to evaluate. Larsen *et al.* (1999) also uses rectangular swept sphere volume as bounding volumes. Such structures are efficient for minimum separation distances and collisions.

In Guéziec (2001), the author uses progressive meshes Hoppe (1996) to get a good approximation of the closest point early on. A progressive

mesh is a hierarchical structure which provides different level of details for the original mesh. The Meshsweeper algorithm uses the distance from the coarse mesh and refines the results based on an early guess. The advantage of this method is that it can provide an early bounded estimate of the distance. However, the implementation can be quite complicated, and its efficiency for exact distances compared to other structures has not been evaluated. Progressive meshes could also lead to unexpected results depending on the topology of the object.

In Lee *et al.* (2013), an octree was used to store the closest element indices instead of the typical surface elements. The tree is called a primitive tree. A query point therefore does not prune out branches, but instead finds the smallest cell it lies in. This reduces the number of node traversals for ambiguous points which are close to the medial axis.

Point membership for meshes In the general case, the distance sign at the given point cannot be computed directly using the surface normal because the mesh surface has C^1 discontinuities. The sign is also ambiguous, if the mesh is non-manifold. The common issues with non-manifold meshes include self-intersections, degenerated faces, inverted normals, cracks and holes.

The sign can be calculated using scan conversion, as detailed in Kaufman (1988). The sign is computed on the lattice in a separate pass. The process is very similar to the scan line rasterization for 3D. The scan conversion is resilient to self-intersections and inverted normals but cannot handle cracks. Unfortunately, this technique is limited to discrete grids.

The most natural extension of this techinique for continuous evaluation at an arbitrary point is ray casting (Requicha 1996). The number of intersections between the mesh and the ray which starts at the query point and is cast towards the outside of the initial polygonal mesh can provide a sign. If the number of intersections is odd, then the query point is inside, otherwise it has an even number of intersections, and is set to be outside the solid bounded by the mesh. Such a solution often suffers from numerical issues. In particular, when a ray lies in the plane

of a triangle it intersects, the ray could either cross to the other side, or not. Ray casting also suffers from self-intersections and holes, but tolerates inverted normals. The evaluation of the number of intersection can also be time consuming for large meshes.

The most convenient method for determining the sign for a continuous signed distance function is that of the angle-weighted pseudo normals (Baerentzen and Aanaes 2005). Edges and vertices have pseudo normals assigned, which can be used to accurately deduce the sign, provided that the closest feature on the surface to the point $\mathbf{p}$ is available. This method can only work with manifold meshes, otherwise sign flips will occur. It is an efficient solution but lacks robustness.

Another solution to provide a more robust sign was introduced by Rossignac *et al.* (2013) for self-intersecting surfaces. The objective was to provide a real-time trimming solution for rendering. This means that it will correctly choose between union (inside) or intersection (outside) depending on the type of intersection. The normal orientation in this case is crucial.

More recently, Jacobson *et al.* (2013) proposed a robust method to find point membership from polygon soups. It generates the correct sign if the object is manifold and a well behaved one if the object is non-manifold. It behaves consistently regardless of the orientation of the object making this solution robust and consistent. This method relies on a generalization of the winding number. The winding number in a plane is the number of times a curve circles counter clockwise around a point. Its formulation is not limited to polygonal meshes only, but the extension to NURBS surfaces is not straightforward. The major drawback of this solution is its linear complexity. Since scalar field based applications require a large number of samples (several million samples even for low precision), linear complexity per query is not practical.

2.3.2.2 Non-uniform rational B-Spline solids

There are two main ways to find the closest point from the given point $\mathbf{p}$ to a NURBS curve or surface: tessellation based and numerical root finding. An analytical solution can be achieved for low order (quadratic) curves and surfaces relatively easily (Gludion 2009). However, if the curve is cubic or of a higher degree, then numerical solutions need to be employed to avoid numerical instabilities (Piegl 1991).

Numerical solutions rely on the following equations:

$$(\mathbf{C}(t) - \mathbf{p}) \cdot \mathbf{C}'(t) = 0 \tag{2.9}$$

where C is the parametric curve, $\mathbf{p}$ is the query point, and t is the parametric coordinate of the closest point.

And similarly, for surfaces:

$$(\mathbf{S}(u, v) - \mathbf{p}) \cdot (\frac{\partial \mathbf{S}}{\partial u} \times \frac{\partial \mathbf{S}}{\partial v}) = 0 \tag{2.10}$$

Note that in the case of NURBS, it is possible to have sharp edges, by having multiple knots. In that case, the curve or surface has to be subdivided (split) and the resulting surfaces have to be treated separately.

A popular method for the point projection is the Newton-Raphson algorithm, which is detailed in Piegl and Tiller (1997). This algorithm iteratively approaches the root of a function by using derivatives. It requires an initial guess, which has to be close to the root, and a close search range. Unfortunately, this method suffers from numerical instabilities and convergence issues. In Ma and Hewitt (2003), the surface is first subdivided into Bézier patches, and the Newton-Raphson algorithm is applied to the patches which could yield a minimum distance. Dyllong and Luther (2000) allows for further subdivision, and includes interval arithmetic to provide more robust results.

To accelerate the search, the convex hull property and subdivision can be used to create a tree structure. Most such trees rely on axis

aligned bounding boxes or oriented bounding boxes (Gottschalk *et al.* 1996). Such bounding volume hierarchies have been successfully ported on the graphics processing unit (GPU) (Krishnamurthy *et al.* 2011). Other bounding volumes have been examined such as rectangular swept sphere volumes (Larsen *et al.* 2000) or Coons patch swept sphere volumes (Kim *et al.* 2011). A patch is approximated by a Coons patch, and the error is used for the radius of the swept sphere. The Coons bounding volume hierarchy creates tighter bounds and fewer nodes. To evaluate the distance from a point to the node, a tetrahedron is used as a convex hull of the Coons patch. The distance to a tetrahedron is trivial, and the radius of the sphere (the approximation error) can be removed from the distance to the tetrahedron.

Bounding volume hierarchies focus on the spatial arrangement, but cone hierarchies (Johnson and Cohen 2001) can be used to find the roots and provide a tight parametric search range as in Johnson and Cohen (2005). The work from Johnson and Cohen (2005) is extensible to surfaces, but requires particular care for the boundary curves. Cone hierarchies rely on spheres for spatial bounding, and on cones for tangent bounding. Such a structure provides a reliable way of culling out any part of the curve or surface, which cannot satisfy equations 2.9 or 2.10. To build the cones, this method relies on the convex hull property, and the curve hodograph. The curve hodograph convex hull contains all the tangents of the curve.

Despite all these efforts, numerical searches are still slow. While their accuracy can be bounded, an exact result cannot be guaranteed. With this in mind, tessellation of the surface becomes a viable solution and is often preferred (Dyllong and Luther 2000). Tessellation of a curved surface can only approximate its smoothness, but the error can be bounded. A distance to a polygonal mesh can then be employed. Adaptive tessellation is often based on a divide-and-conquer strategy (Cook *et al.* 1987). Abi-Ezzi and Shirman (1991) and Chhugani and Kumar (2001) provide tessellation strategies which can ensure the quality of the approximation in the view space, or for any affine transformation. These methods rely on the bounds on the second derivatives as described in

Filip *et al.* (1986). The work by Fisher *et al.* (2009) also focused on generating a crack free mesh. Cracks usually occur at boundaries where the tessellation level suddenly changes.

2.3.3 Main applications of distance fields

Distance fields have found many applications in robotics, physical simulation, animation, modelling and image processing. Descriptions of more applications can be found in Erleben and Dohlmann (2008) and Jones *et al.* (2006).

2.3.3.1 Collision detection

Distances have been used to help with collision detection for a long time. In Cameron *et al.* (1986), the distance between two convex object is calculated to determine if a collision occurred. Lin and Canny (1991) improved the efficiency of the distance computation by reusing the results from the previous frames. Lin and Gottschalk (1998) provide a more in-depth survey of the methods for computing distances between objects for the purpose of collision detection of rigid bodies.

Unlike the distance between two objects, distance fields can provide more information about the collision normals, and can be used to solve more complex problems. In Fuhrmann *et al.* (2003), distance fields are used to perform collision detection between a rigid body and a dynamic object (cloth, hair, fur). The rigid body is approximated by a discrete distance field, and the particles of the dynamic object are tested against the field for proximity (allowing for a threshold). The gradient of the field provides the normal vector required for the collision response.

Discrete fields were also used to maintain non-penetrating flexible bodies in Fisher and Lin (2001). A tetrahedral mesh is generated for the original object, and each node in the 3D mesh is given a distance value. When the object is deformed, the distance at the nodes can be approximated. If a self-intersection occurs during animation, the

distance values at the nodes can provide the penetration distance and the first and second derivatives which are required for penalty based methods.

In Krishnamurthy *et al.* (2011), the authors performed clearance distance queries by first finding potentially close surface pairs isolating the surfaces to be tested. Then a minimum distance between two surfaces was run against every pair. The distance between two surfaces can be computed by an iterative process using the closest point from each other. This procedure should converge if the parametric bounds on each surface have been reduced to small enough regions.

2.3.3.2 Sweeping

Sweeping a solid along a trajectory can be a useful tool in Computer-Aided Design (CAD). The swept object represents the union of the object samples at all the positions along the curve. Some of the applications include removing (through the set-theoretic subtraction operation) parts of an object that could obstruct a mechanical part from moving, or checking for collision for path planning or robotic arm movements. Because of the complexity of the problem, its simpler formulation uses a planar cross-section template which moves along the curve orthogonally to the tangent on the curve. This problem has been solved with varying degrees of success through the use of distance fields or distance-like fields.

In Schroeder *et al.* (1994), a voxel representation of swept surfaces and volumes is presented for any object for which a distance field can be evaluated. This process performs several unions of the original model at different positions over the voxel lattice. The step size along the curve is bound to ensure an accuracy relative to the voxel size. In Pasko *et al.* (1996), the sweep volume is created by finding the global extremum of the sweep volume function. The sweep volume function is the infinite union of the object samples at every affine transformation it passes through. The function is approximated using polynomial extrapolation in-between sample positions. Schmidt and Wyvill (2005) provide an analytical so-

lution to sweeping a 2D template along a curve. A discrete approximate smooth distance field of the template is generated first. Then, for any point in space, the extrema of the distance to the curve function (see equation 2.9) and the local frames are used to generate a set of 2D point coordinates. The sweeping function is the union of the distance field values at those coordinates. This solution introduces C^0 discontinuities because the number of cross sections of the union can change suddenly, and the field values can vary greatly.

2.3.3.3 Modelling

Offset surfaces are trivial to achieve with distance fields (Farin 1987), although the offset cannot be applied iteratively since the resulting field after the first offset is not a distance field to the offset surface. Combined with set-theoretic operations, this allows for the creation of exact one-sided shells. Constant radius blending can also be achieved using distance fields as presented in Rossignac and Requicha (1984). As a subset of the Function Representation, signed distance fields have numerous applications, some of which are mentioned in Pasko and Adzhiev (2004). However, the C^1 discontinuities rule out smooth blending unions and intersections.

Often modelling operations applied to distance fields do not yield distance fields as a result. Some research was conducted in this area for simple set-theoretic operations in Fayolle and Pasko (2010).

2.3.3.4 Rendering

Geometric objects defined by distance fields can be rendered more efficiently than iso-surfaces of arbitrary scalar fields. Hart (1996) introduced sphere-tracing to improve on ray-marching. In ray-marching, values are taken at constant step along the ray until the iso-surface is crossed. A binary search can locate the iso-surface more accurately. With sphere tracing, the step size is set equal to the distance to the surface. This allows for larger steps when far from the surface. As the search approaches

the surface, the step size is reduced. It is not necessary to have an exact distance function, but the function must provide a value which is less or equal to the distance. In Quilez (2008), the author uses iso-surfaces to quickly generate a procedural world, and mimics global illumination (ambient occlusion, soft shadows, and later on sub-surface scattering) using the distance properties of the function (Evans 2006). The ambient occlusion can be computed by sampling a line segment normal to the surface, and comparing the actual values with the expected values.

Distance fields can also be useful in two dimensions. In Loop and Blinn (2005), shapes (alphabetical characters) are defined as a closed set of Bézier curve contours. This solution provides compact and resolution independent images. Anti-aliasing is achieved by making the opacity a function of the distance to the boundary. The thickness of the band which alters opacity is directly dependent on the screen pixel size. This narrow band is set to be $\frac{1}{2}$ a pixel.

In a similar fashion, Green (2007) used signed distances to render crisp text and binary images. However, they used a discrete field (a 2D image) which could be loaded to the graphics processing unit easily, and did not require Bézier curves. First, an image is created which classifies each pixel as inside or outside the shape. A distance transform is then applied which computes the shortest distance to a pixel of an opposite value. Bi-linear interpolation is used to reconstruct a continuous function, and a simple step function allows us to filter what should be rendered. It also introduces various effects such as embossing.

2.3.4 Conclusions on distance fields

Distance fields have many applications in various areas. Since the needs of these areas are different, several formats for distance fields have been introduced. First, the original formulation is a continuous scalar field defined in space, which provides an exact distance. Signed distances can be used to represent volumes, where the sign is used to indicate the interior and exterior of the volume. The most common and popular

format however is a discrete set of voxels, which contains either an exact or an approximate value. Regardless of what they contain, voxels are approximations of the distance field function in-between the cells and outside the boundaries of the voxel grid. Therefore, exact continuous distance fields are more suitable for a function representation. Exact distance fields preserve the exact shape unlike discrete solutions and the sampling far from the mesh is more fitting to the overall shape of the object. In this work, we will refer to exact continuous distance fields simply as distance fields.

In the context of heterogeneous volumetric modelling, we cannot rely on voxels and a continuous function is needed, which can preserve the exact input boundaries for the material *feature*. At the same time, smoothness of the exact distance function can be problematic. Alternatives such as L_p-distances fulfil both continuity and boundary requirements, however the computational cost of L_p distances can be an issue. Furthermore, signed L_p distances are restricted to meshes only. This section has shown that a field which respects the boundary condition as well as the C^1 continuity requirement while maintaining a low complexity is essential but does not exist.

Aside from signed distance fields, other scalar fields have been proposed, and interior distances seem particularly relevant to heterogeneous volumetric modelling. Typical distance field evaluation algorithms assume an unobstructed path from the point to its closest point on the surface. This can be problematic for some operations.

2.4 Geometric operations

In this section, we focus on some essential operations which will be used in the following chapters. While some references are given for the boundary representation, the main focus is on the function representation of those operations. Set-theoretic operations are commonly used in more complex operations. However, there are more than one way of performing these operations in FRep. The first subsection introduces various

ways present in literature and draws conclusions on the merits of each. Metamorphosis has been studied thoroughly and this survey will serve as a literature review for section 3.5 in which metamorphosis is used selectively to mix and match parts of different models. Metamorphosis is also an important part of section 3.4 where the properties of two objects are interpolated across time. Finally, the survey of microstructures is relevant since many applications rely on microstructures to showcase the use of varying attributes across space.

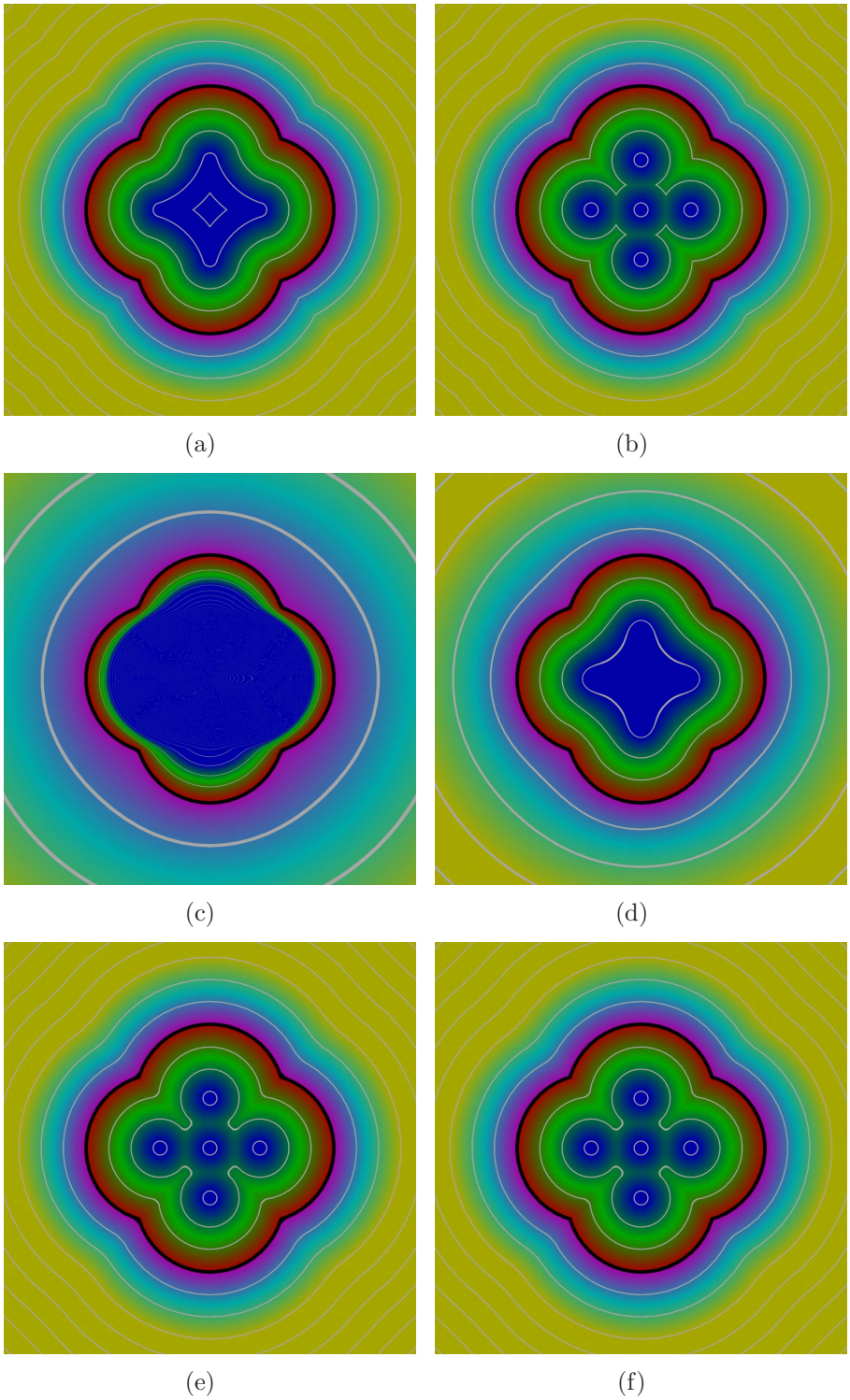
2.4.1 Set theoretic operations in FRep

Solids defined by the FRep can easily undergo set theoretic operations. There are several ways of performing those operations (Ricci 1973; Rvachev 1982; Shapiro 1991; Pasko *et al.* 1995; Fayolle *et al.* 2008) yielding the same shape, but with various effects on the scalar field (compact support, continuity, distance properties). Set theoretic operations are particularly useful for modelling mechanical parts and man made objects.

For two given objects defined by the FRep functions f_1 and f_2, the set-theoretic operations (of union, intersection and subtraction) can be defined with R-functions. f_1 and f_2 are continuous real valued functions of point coordinates, where the sign is used to indicate whether the point is inside (positive) or outside (negative). The surface is the zero-level point set. R-functions are real-valued functions for which their sign only depends on the signs of their arguments. R-functions naturally extend Boolean functions to real valued functions. The common operators are defined as follows:

- $f_3 = f_1 \vee f_2$ defines a union
- $f_3 = f_1 \wedge f_2$ defines an intersection
- $f_3 = f_1 \setminus f_2$ defines a subtraction

There are many applications for R-functions, and therefore, the choice of the right tool depends on the applications. There are a number of properties which can be desirable for R-functions. Some of these properties



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

Figure 2.13: *An object made of 5 discs using unions. (a) shows the actual distance to the surface of the union, reference (b) shows* $\vee_1$*, (c) shows* $\vee_0$*, (d) shows* $\overset{0}{\underset{2}{\vee}}$*, (e) shows SARDF union, (f) shows the adaptive rounded union*

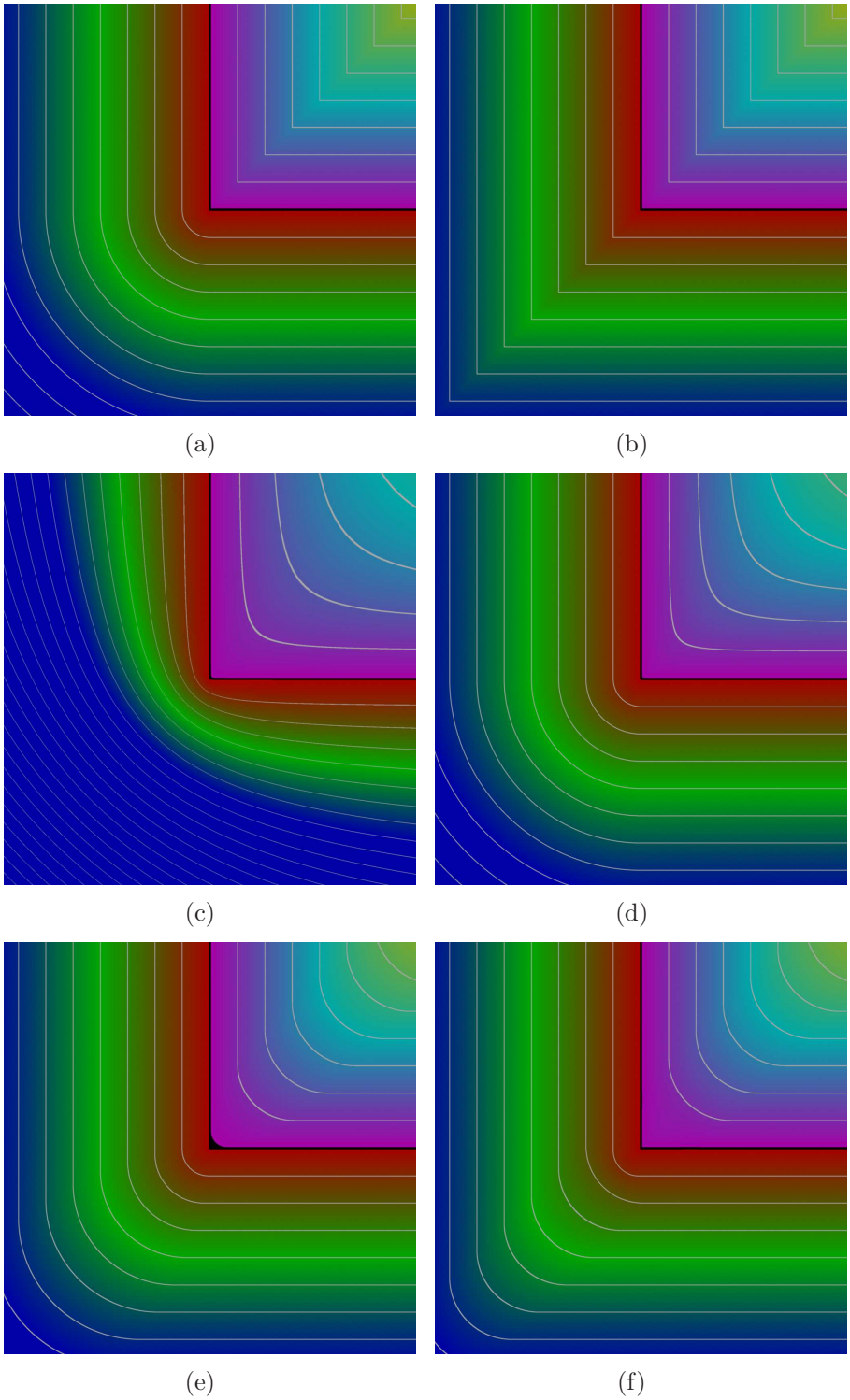


Figure 2.14: *Different R-function unions of two planes. Black lines represent the zero-set surface, and the white lines are uniformly spaced iso-levels. (a) shows the actual distance to the surface of the union, reference (b) shows* $\vee_1$*, (c) shows* $\vee_0$*, (d) shows* $\overset{0}{\underset{2}{\vee}}$*, (e) shows SARDF union, (f) shows the adaptive rounded union*

are discussed more in depth in Shapiro (2007).

- The distance properties. An exact distance can be essential for applications like offsets, but usually, only approximate distance properties are necessary. Distance properties can be defined as follows: if $d(a) \leq d(b)$ then $f(a) \leq f(b)$ where d is the exact distance function, and f the function to be judged. The loss of distance properties often leads to unstable fields and counter intuitive behaviours for some operations.

- C^1 continuity. As highlighted in the previous section, C^1 continuity is required by many applications for both the geometry of an object and its attributes. C^1 continuity is contradictory to the nature of the exact distance function. With this in mind, it is important to distinguish two types of C^1 continuous R-functions, those which bound the smoothing area and can be controlled by the user (*adaptive*) and those which cannot (*fixed*).

- Symmetry. All R-functions are associative and commutative at the surface level. But most fail to carry this property over the entirety of the scalar field. This can lead to several issues and counter intuitive behaviours. Symmetry is critical to preserve distance properties, and are necessary for predictability. For instance, the order in which unions are performed before an offset should not impact the resulting shape.

- Culling. Often numerous unions are performed to create the bulk of an object. In this case, an efficient solution is to cull out some of the primitives based on their positions and approximate the distance, if the R-function allows it.

Figure 2.13 and figure 2.14 are used to illustrate the differences between a few R-functions. The colours indicate the values of the scalar field. The black lines represent values close to the zero-level set. The white lines indicate other iso-levels. Blue values indicate large positive values, and yellow large negative values. For both pictures, the first subfigure (figure 2.13a and figure 2.14a) are the exact distance fields to the

surface. In figure 2.13, a shape is built using five discs laid symmetrically. In figure 2.14, a union between two perpendicular planes is performed.

The most popular solution is to use the minimum and maximum functions (Ricci 1973) (shown in figure 2.14b).

- $f_1 \vee_1 f_2 = \max(f_1, f_2)$
- $f_1 \wedge_1 f_2 = \min(f_1, f_2)$
- $f_1 \setminus_1 f_2 = \min(f_1, -f_2)$

These operations have C^1 discontinuities whenever $f_1 = f_2$. Also, they do not approximate the distance well. The lower left quadrant in figure 2.14b clearly shows the deviation from the distance field shown in figure 2.14a. Their advantage is that if the smallest value can be identified before sampling, only one of the two functions need to be evaluated. This is possible if we are dealing with distance fields and use the bounding boxes as an initial guess.

Another class of R-functions which may only introduce C^1 discontinuities at the intersection of the two surfaces was presented in Rvachev (1982) (shown in figure 2.14c).

- $f_1 \vee_0 f_2 = f_1 + f_2 + \sqrt{f_1^2 + f_2^2}$
- $f_1 \wedge_0 f_2 = f_1 + f_2 - \sqrt{f_1^2 + f_2^2}$
- $f_1 \setminus_0 f_2 = f_1 - f_2 - \sqrt{f_1^2 + f_2^2}$

However, such functions lose distance properties (shown in figure 2.14c). If used numerous times, the field becomes chaotic. These functions also lack the associative and commutative properties, which implies that the order of the application of an operation changes the field. A symmetric shape gives an asymmetric field (see figure 2.13c). Regardless of the fields, both arguments need to be evaluated, therefore culling one of them is not possible.

Rvachev (1982) also introduced another class of R-functions called the $\underset{2}{\overset{0}{R}}$ class. The cross section of the union of two planes using this class of R-functions is shown in figure 2.14d. The intersection formulation is

shown below:

$$f_1 \overset{0}{\underset{2}{\wedge}} f_2 = \begin{cases} f_1 f_2 ({f_1}^2 + {f_2}^2)^{-\frac{1}{2}} & \text{if } f_1 > 0 \text{ and } f_2 > 0 \\ f_1 & \text{if } f_1 \leq 0 \text{ and } f_2 \geq 0 \\ f_2 & \text{if } f_1 \geq 0 \text{ and } f_2 \leq 0 \\ (-1)^{2+1} ({f_1}^2 + {f_2}^2)^{\frac{1}{2}} & \text{if } f_1 < 0 \text{ and } f_2 < 0 \end{cases} \tag{2.11}$$

These functions have very good distance properties while also maintaining C^1 continuity everywhere but at the surface intersection. These functions are also commutative and associative, making them more suitable for large FRep models which contain numerous operations (see figure 2.13d). These R-functions seem to approximate the distance function well, as shown in figure 2.13a. Culling is also possible if the sampling point is inside one solid, and not the other. This means *some* culling is possible.

Another class of R-functions was introduced in Fayolle *et al.* (2008) called Signed Approximate Real Distance Functions (SARDF) which maintains C^1-continuity everywhere except where the surfaces intersect. These functions allow for an adaptive smoothness where the smoothness is controlled by the users. This means that the function can closely match the operators from Ricci (1973), or be as smooth as the $\overset{0}{\underset{2}{R}}$ functions. The function also bounds the distance approximation error given that their arguments are distance approximations too (shown in figure 2.14e). Similarly to the $\overset{0}{\underset{2}{R}}$ class functions, both functions do not need to be evaluated in some cases.

In ImplicitCAD (2013), another function for *rounded unions* (similar to blended unions) was introduced. The roundness is controllable and blends the objects together.

$$f_1 \wedge_{rounded} f_2 = \begin{cases} \min(f_1, f_2) & \text{if } |f_1 - f_2| \geq r \\ f_2 + r \cdot \sin\left(\frac{\Pi}{4} + asin(\frac{f_1 - f_2}{r\sqrt{2}})\right) - 2 & \text{if } |f_1 - f_2| < r \end{cases} \tag{2.12}$$

In this equation, r is the roundness radius. In this form, this function is not an R-function. However, by varying the value of r in function of

Method	Distance	C^1 continuity	User control	Symmetric	Culling
R_1	Low	No	No	Yes	Yes
R_0	Low	Yes	No	No	No
$\overset{0}{R}_2$	Good	Yes	No	Yes	Partial
SARDF	Good	Yes	Yes	No	Partial
Adaptive rounded	Medium	Yes	Yes	No	Yes

Table 2.1: *A comparison R-functions desirable properties*

its distance to the surface, it can become an R-function. The roundness is set to zero at the surface, and is increased as the distance to the surface increases. Its behaviour is very similar to the SARDF. SARDF are shown in figures 2.14e and 2.13e, while adaptive rounded unions are shown in figures 2.14f and 2.13f. This function is advantageous because the smoothing area is bounded easily which can be useful for multiple unions by evaluating fewer primitives and has a simpler implementation than SARDF.

The table 2.1 summarizes the properties of the various R-functions presented in this section. The distance column provides an evaluation of how well the function approximates the distance function. The C^1 continuity column deals with the C^1 continuity of the scalar field everywhere but at the surface. The user control column shows which functions allow users to adjust the smoothness. Symmetry is important so that the order of operations does not produce different scalar fields (e.g. $A \vee B$ should be equivalent to $B \vee A$). Finally, some R-functions do not require both values to be evaluated given some conditions. If we have two distance fields to union, and one object is closer than the other to the sampling point, then it may be possible to only sample the closest object.

Apart from the R_0 functions, all those R-functions can be useful depending on the desirable properties of the resulting field. The R_1 class is extremely efficient, but lacks the C^1 continuity which is so critical for many applications. The only proven commutative and associative R-function is the $\overset{0}{R}_2$ class but its smoothness cannot be controlled and therefore it is difficult to control the smoothness of the field. SARDF and adaptive rounded functions provide good distance approximations, and

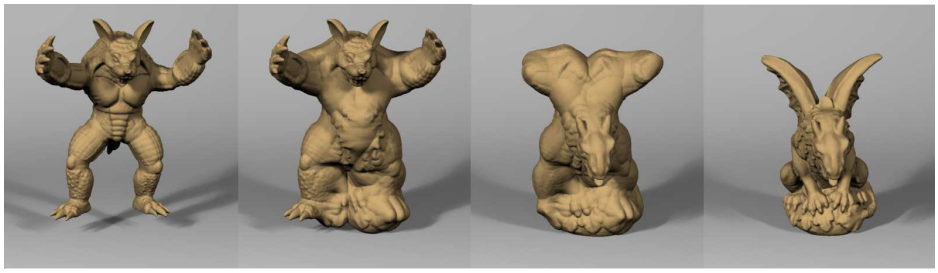


Figure 2.15: *Metamorphosis from an armadillo to a gargoyle using Pasko* et al. *(2004)*

allow for controlled smoothness. However, both $\overset{0}{R}_{2}$ and SARDF require both values f_1 and ${}_f2$ in most cases, while R_1 and adaptive rounded functions can be evaluated without necessarily evaluating both values.

In the following chapters, the appropriate R-function will be selected based on the desired properties. R-functions are used in section 3.3, section 3.4 and in a number of applications in chapter 4. In each, distance properties, continuity and culling are the main concerns.

2.4.2 Metamorphosis

Metamorphosis has often been used in animation, but also in modelling as an artistic tool or a shape finding method. In Adzhiev *et al.* (2005), group metamorphosis is used to generate a number of sculptures. The metamorphosis work by Dutch artist M.C. Escher inspired Pasko *et al.* (2011b), in Formnation (2014), a breeding process was applied to chairs using metamorphosis. In McLoughlin *et al.* (2016), a simple metamorphosis tool was provided to children with disabilities to create original objects which were later 3D printed.

In computer graphics and animation, metamorphosis is a continuous transformation from one shape (*source*) into another (*target*) through intermediate shapes. In Lazarus and Verroust (1998), at least three approaches were distinguished: based on polygonal meshes, based on scalar fields and on conversion methods.

Most of the methods based on polygonal meshes assume that the given

shapes have matching genus and number of components, and that a correspondence between the meshes can be established (through indices or texture coordinates). Meshes that have different topology can be used in metamorphosis, however some additional knowledge about topological changes and additional user intervention are required in this case (DeCarlo and Gallier 1996; Takahashi *et al.* 2001).

Metamorphosis between two models represented by FRep (Pasko *et al.* 1995) or discrete scalar fields can be performed regardless of the objects topology or shape alignment (Hughes 1992; Yang and Jüttler 2007). He *et al.* (1994) performed the interpolation in the wavelet domain to improve the transition smoothness. However, there is almost no user control which can lead to unsatisfactory results. For instance, the linear interpolation of the values of the scalar fields representing the shapes has a simple formulation but can produce poor results or even fail to produce intermediate shapes (Pasko *et al.* 1995; Cohen-Or *et al.* 1998). Turk and O'Brien (1999a) proposed a more sophisticated approach based on interpolation of surface points with assigned time coordinates using radial basis functions in 4D space. This method can handle non-aligned surfaces with different topologies. However, the initial shapes (source and target) need to be sampled and rely on radial basis functions, which can be time-consuming to evaluate. Another method of shape metamorphosis called space time blending was introduced in Pasko *et al.* (2004) (see figure 2.15). This method does not require shape alignment, however the user can only affect the entire metamorphosis process rather than specific features of the shape and its parameters can be difficult to adjust correctly.

More user control can be achieved by restricting the class of objects. Thus, in Wyvill (1993) some control over the metamorphosis of skeletal soft object was achieved by cellular matching or hierarchical matching and the metamorphosis was performed by interpolating the positions of the skeletons and the field intensities. The main drawback of this method is the requirement for skeleton matching for two objects which is similar to the shape matching problem defined for polygonal models. Later in Galin *et al.* (2000) this limitation was partially lifted, however

both shapes need to have a similar structure for the interpolation to be possible.

Lerios *et al.* (1995) introduced a voxel-based metamorphosis with improved user control, where the user can define object features such as points, rectangles and boxes. Chen *et al.* (1996) also improved the user control by providing a similar solution based on a number of pairs of discs used to mark correspondences between the objects. The authors of Lerios *et al.* (1995) and Chen *et al.* (1996) were inspired by the Beier and Neely (1992) approach, which allows for creating smooth transitions between two images. This technique allows the user to define pairs of corresponding features in the source object and the target object and to interpolate between the shapes of the source model and the target model by using the information about these features. Features can be defined as a frame with a bounding box, providing more user control than other existing techniques. However, this technique heavily relies on the linear interpolation between shapes which occasionally does not produce acceptable results.

Group metamorphosis, where more than two objects are interpolated, was addressed for limited groups of objects with simple weighting schemes such as barycentric coordinates for three objects (Adzhiev *et al.* 2005) and the bilinear interpolation for four objects (Fausett *et al.* 2000).

As for transformation of volumetric attributes during the metamorphosis, two groups of solutions are available. Particle systems are used to follow surface properties and flow from the *source* to the *target* object. Drastic changes in topology tend to provide unrealistic or undesirable effects. The work of Smets-Solanes (1996) improved on the above mentioned particle system technique to get good results for animated implicit surfaces. A set of points on the surface of the initial shape (virtual skin), parameterised in 2D, is transformed using different vector fields defining the particle velocities. Several effects such as clay style texturing can be achieved and splitting or merging objects are handled correctly. The vector fields are however difficult to create and they often exhibit case specific problems. The work of Tigges and Wyvill (1998) extended tex-

turing through the use of particles to deal with gradient discontinuities that may arise with constructive models and distance functions. The force applied to the particle is a weighted sum of forces. The gradient vector field is used to repel the particle, but the support shape also acts as an attractor. Several operations are also extended to blend textures and surface materials together.

Alternatively, PDE-based methods track surface changes. In Dinh *et al.* (2005) a method was introduced to track surfaces with corresponding attributes in time by solving PDEs. This solution gives consistent results with any topological changes including splits and holes. In the work of Bojsen-Hansen *et al.* (2012), the authors used a similar method to track properties (such as colours and displacement information) during the shape transformation process in morphing or in fluid simulation with heterogeneous fluids. This method is able to handle drastic topology changes. However, solving the PDEs is time consuming and makes this solution impractical for real time applications, in long animation sequences or when applied to large meshes. Both groups of solutions are limited to surface attributes (usually colours).

Metamorphosis is an important part of the work presented in section 3.5 and in section 3.4. In section 3.5, it is the starting point to generate shapes from parts of various objects. Section 3.4 deals with the problem of the interpolation of volumetric properties between two objects during metamorphosis.

2.4.3 Microstructures

Microstructures are internal spatial geometric structures with size of detail orders of magnitude smaller than the overall size of the object. They could be used to reduce the amount of material in object fabrication, or to change the composition of an object to allow for other properties such as thermal insulation. Such structures can be classified as periodic porous structures or irregular porous structures (Giannitelli *et al.* 2014).

Boundary representations are poorly suited for handling microstruc-

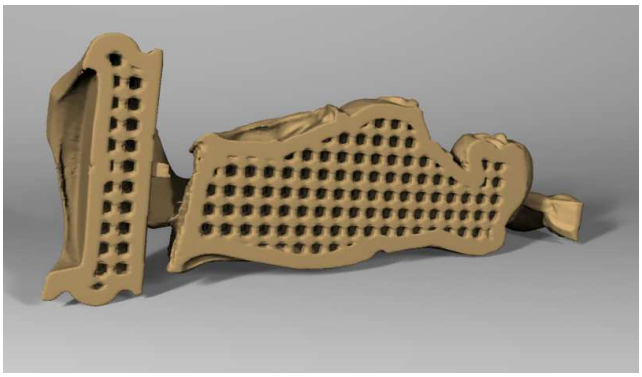


Figure 2.16: *Simple lattice microstructures inside a Buddha figurine*

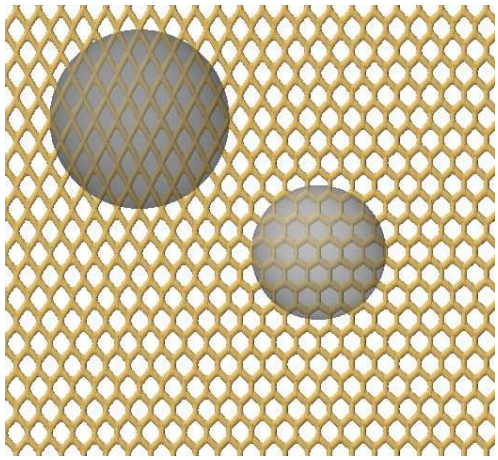
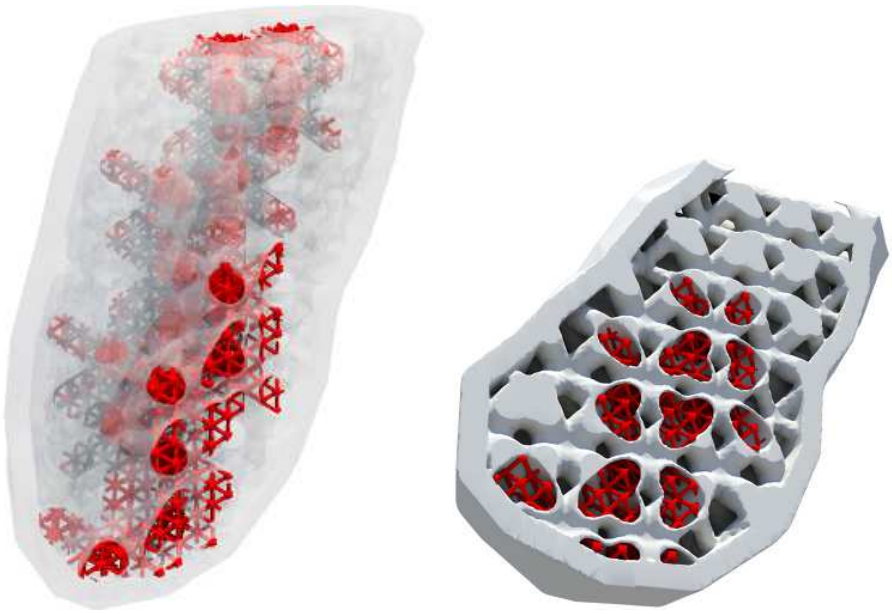


Figure 2.17: *Microstructure parameters are interpolated across space. Image from Fryazinov* et al. *(2013)*

tures. First, microstructures can require a huge number of polygons, even for simple lattices. The model size becomes even bigger, if the lattice has smooth features such as blended intersections. If smooth surfaces are used, then accuracy also becomes an issue. Finally, handling microstructures with various other objects can lead to self-intersections, making the object non-manifold. Having some of the attributes of the lattice change across space (based on the distance to the surface, or using data from a numerical simulation, for instance) can lead to even more issues. Crafting unit cells can also be tedious. This led Cheah *et al.* (2003a,b) to create template parametric libraries of cellular structures. However, the function representation can easily fill an arbitrary object with various microstructures without having any of the above issues.

In Pasko *et al.* (2011a) a lattice microstructure is modelled where the



(a) (b)

Figure 2.18: *Multi-scale microstructures, where microstructure can contain microstructures themselves. Image courtesy of Daniel Büning*

intersections between rods are blended, as well as the connections between the microstructures and the shell of the object (see figure 2.16). Additionally, the parameters of the microstructures, such as thickness and spacing, are controlled by the distance to the boundary. The structure can also undergo space mappings (such as twisting, bending or tapering). Arbitrary microstructures can be constructed using cell replication. The parameters can also be mixed with noise functions allowing for natural looking porous objects. In Fryazinov *et al.* (2013), several improvements are presented. The transfinite interpolation (Biswas *et al.* 2004) is applied to the parameters of the microstructure. Attributes include a morphing parameter between one type of cell and another (see figure 2.17). Some features of the microstructures can also be made of other microstructures, and further down several levels (see figure 2.18). The function representation is capable of applying Boolean operations to such objects regardless of their complexity, allowing the user to have surface microstructures by intersecting the infinitely repeating structure with a shell of the object. Typical microstructures which can be achieved with such technique are the honeycomb structures and the sea sponge

structures. In Gabbrielli *et al.* (2008); Melchels *et al.* (2010), the authors introduced a list of functions for microstructures in order to reproduce natural bone structures. Additionally, the user can select a target volume fraction, providing a clear parameter to control the object weight. Such microstructures rely on the cell replication (tiling), although the cell attributes can change across space. Schumacher *et al.* (2015) also used tiling of a structure to change the elasticity of a model by changing the parameters of the microstructures across the object volume. Peytavie *et al.* (2009) procedurally generated rocks using aperiodic tiling in order to break the undesirable repetitiveness of the pattern. This can lead to more natural looking patterns similar to some of the irregular porous patterns introduced in Pasko *et al.* (2011a). The common drawback shared by all the above methods is that they rely on intuitions to provide robustness, or simply on aesthetics.

In material engineering, microstructures such as polymer foam are popular for their robustness and light weight. In Wejrzanowski *et al.* (2013) the Laguerre-Voronoi tessellation is used to reproduce similar structures. The results are compared with foam structures found in nature by the average number of faces per cell. A Laguerre-Voronoi tessellation is a variation of Voronoi tessellations where a weighting is applied per seed. The weighting can be interpreted geometrically as the radius of a sphere. Some computer modelling systems reproduce solid foams (Lal and Sun 2004) by using sphere packing. Sphere packing is the process of fitting as many spheres as possible within a volume. Here the radii of the spheres vary to fill gaps in-between the large spheres.

Heterogeneous object modelling methods are often applied to microstructures to demonstrate the usefulness of varying attributes. Microstructures can have varying properties across an object (such as density, elasticity, robustness). In chapter 4, many applications will show how the microstructure properties can be controlled with the methods presented in chapter 3.

2.5 Conclusions

In this chapter, we reviewed various object representations for heterogeneous object modelling and various approaches to designing heterogeneous objects. There are two main models, evaluated and unevaluated models. While evaluated models is more suitable for numerical simulations, unevaluated models are interesting from a design stand point. In the unevaluated models, feature based methods, which rely on distance fields, have shown promising results. Distance fields were surveyed, and it was shown that they have many beneficial properties.

In this survey, we highlighted a number of methods for heterogeneous object modelling. For design purposes, the unevaluated methods are more appropriate (Kou and Tan 2007). The challenge of unevaluated methods is to make these methods easy-to-use and controllable. Feature-based methods are user-friendly and easily integrated into existing CAD systems. However, they are not as flexible as constructive methods. For instance, feature based methods are not shape aware while constructive methods can be shape aware and provide precise control.

As we have shown, distance fields are widely used for heterogeneous object modelling, and many authors have argued that they were important for heterogeneous object modelling (Rvachev *et al.* 2001; Biswas *et al.* 2004; Fayolle *et al.* 2006). Feature based methods rely on distance fields for numerous reasons. One of these reasons is that distance fields make interpolation in material gaps predictable. Additionally, most geometric features can be represented using distance fields, and any object modelled in traditional CAD systems can be represented by scalar fields.

A number of issues remain however. Distance fields are not smooth, which can be problematic. Exact distances and smoothness are often irreconcilable, however, approximate distances can be used to preserve the predictability (Fayolle *et al.* 2006) and provide smoothness at the same time. Another recurring issue in feature based methods is the issue of shape awareness. Most feature based methods rely solely on the features of the object and do not take into account the shape of the object. Fi-

nally, as explained in Kou and Tan (2007) and Kravtsov (2011), most of the research has been focused on static objects. Therefore, time dependent heterogeneous object modelling has not received a lot of attention.

In this thesis, some of the issues discussed above are tackled. In the next chapter, a smooth approximation of distance fields is proposed in section 3.2. Section 3.3 tackles the issue of shape awareness for feature based heterogeneous object modelling. Section 3.4 provides a method to interpolation two volumetric material distributions of two objects during metamorphosis.

Chapter 3

Distance based heterogeneous volumetric modelling

Heterogeneous volumetric modelling finds applications in many areas such as bio-engineering, multi-material design and fabrication, geology and physical simulations. In the previous chapter we have outlined the state of the art for heterogeneous volumetric modelling and to which extent it relies on distance fields. Heterogeneous volumetric modelling, if considered mainly as object material modelling, can be divided in two major types, composite materials and gradient materials. Composite materials can be defined in several ways and are not as challenging as gradient materials. Composite materials can be seen as a collection of homogeneous parts. For gradient materials, *feature-based* methods are the most prevailing thanks to the easiness of the configuration process. However, several issues, highlighted in chapter 2 showed that improvements could be made in this area. Additionally, some problems cannot yet be solved using the currently available techniques.

In section 3.1, we will discuss a theoretical framework for distance based heterogeneous volumetric modelling to help us solve some of the problems indicated in chapter 2. In the following section, a new smooth, approximate distance field will be introduced to alleviate the issues related to distance field discontinuities in heterogeneous volumetric modelling and shape modelling. In section 3.3, interior distances will form

the foundation of a novel method to interpolate material and other volumetric properties across a shape based on the *feature-based* modelling principle. In section 3.4, distance fields will be used to tackle the problem of the interpolation of volumetric object properties through time as its shape changes. Finally, smooth distance fields will be used to support a complex morphological shape generation technique allowing designers to combine and mix different parts of various objects together.

3.1 General approach

A heterogeneous object is an object composed of diverse materials or possessing other properties varying across its volume. Such properties include colour, material types, temperature, density, structural patterns and others. Heterogeneous objects are common in nature, such as fruits and plants, and some man made objects can be considered heterogeneous objects too. Two types of properties distribution across a volume can be distinguished; *composite* with abrupt changes from one value to another; *gradient* where a property gradually changes in the object. An object may combine both composite and gradient properties.

As mentioned previously, there are several representations for heterogeneous objects, and a few techniques and frameworks have been introduced for heterogeneous volumetric modelling. Some used a hybrid representation to represent the geometry on one hand, and the values of the properties on another hand. Voxels have been popular even though they are limited by their resolution. The constructive hypervolume framework allows to represent both geometry and properties using FRep, and the focus of this thesis is on extending this framework.

In chapter 2, an object in the constructive hypervolume framework was defined by equation 2.3. The object is defined by its geometry and a number of attributes, which may represent physical properties.

$$o = (G, A_1, \ldots, A_i) \tag{3.1}$$

- o is the object
- G is the geometry of the object
- A_i is an attribute of the object

In Pasko *et al.* (2001), the heterogeneous object can be modelled using a function for its geometry, and any number of attribute functions. An attribute function can be a vector function (e.g., for colour) or a scalar real-valued function (e.g., for temperature). An extension is proposed to adapt this earlier formulation to support time variant attributes, and attributes as functions of distances. The model of the heterogeneous object can be considered a vector function. The equation 2.3 is extended as follows:

$$\Gamma(\mathbf{p}, t) = \big(F(\mathbf{p}), S_1(\psi_1(\bar{\phi}), t), \ldots, S_i(\psi_i(\bar{\phi}), t)\big) \tag{3.2}$$

where

- $\mathbf{p}$ is a point in Euclidean space
- t is the time instance
- Γ is the model of the heterogeneous object, a vector function. Each component of Γ is a model of the corresponding component in the tuple of equation 3.1.
- F is a real-valued function of point coordinates which represents the object geometry G
- S_i is a function representing an attribute, which depends on a number of distance functions and time. Each function S_i represents an attribute A_i
- ψ_i is a function of a number of distance functions
- $\bar{\phi}$ is a vector of distances

$$\bar{\phi} = (\phi_1(\mathbf{p}), \phi_2(\mathbf{p}), \ldots, \phi_i(\mathbf{p})) \tag{3.3}$$

where ϕ_i is a distance to either a space partition boundary, or to a feature

such as a line, a point or a curve.

Equation 3.2 defines the procedure to evaluate the object at the given point and time instance, we evaluate F, and if it is positive, then for each attribute, we evaluate the functions S_i. Equation 3.2 differs from equation 2.3 because attributes are now functions of distances and time. In simpler terms, the focus of this work is on attribute functions of distances and time. While some attributes are not functions of distances or time, we will show that numerous attributes can be expressed by functions of distances. Aside from the more physically based methods, distances can provide more intuitive fields for users to work with. It can be relevant when modelling geometric objects (e.g., blending them) but usually, the quality and distance properties of the scalar field have a limited impact on the intuitiveness of the operations (e.g. Boolean operations, twist). However, heterogeneous volumetric modelling and gradient properties rely solely on the scalar field, and therefore distance fields become imperative in order to avoid unexpected results as discussed in chapter 1.

If ϕ_i is a function with distance properties, ϕ_i assumes a zero value on the surface of the object, a positive value inside the object and a negative value outside it. Also, the absolute value of the function grow with the distance of a point from the surface. Thus, if $d(\mathbf{p}_1) < d(\mathbf{p}_2)$, then $\phi_i(\mathbf{p}_1) < \phi_i(\mathbf{p}_2)$ where d is the distance function. Additionally, modelling the object with functions of time allows for time-dependent transitions of attribute distributions from one value to another or helps express certain properties as functions of time. For instance, temperature is a function of time and distance to a source.

On the basis of the introduced general approach to deal with attributes expressed by functions of distances and time, several long-standing heterogeneous volumetric modelling problems were selected and in this chapter we show how they can be addressed from the proposed theoretical point of view. As stated in chapter 2, the continuity of the functions $S_i(\psi_i(\bar{\phi}), t)$ is not essential. For instance, composite materials are inherently expressed by discontinuous functions at the boundary of two

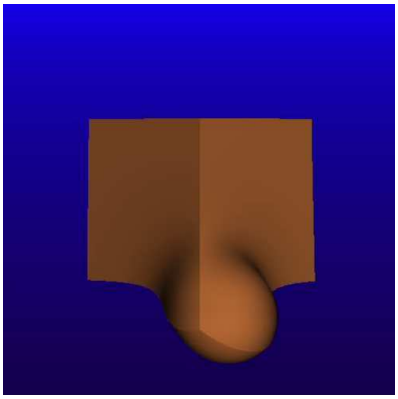


Figure 3.1: *The blending union between a cube and a sphere can cause a crease to appear. This is due to the C^1 discontinuities of the cube function, built using min/max. Image courtesy of Pierre-Alain Fayolle*

materials. However, if such attributes are to be used for modelling purposes afterwards (e.g., to model microstructure thickness controlled by a function), C^0 function continuity is required to maintain a valid FRep object.

In most cases, a C^1 continuous ϕ function is necessary for a smooth volumetric attribute. Smooth transitions for volumetric attributes are important to avoid the stress concentrations that can be caused by C^1 discontinuous functions (Rvachev *et al.* 2001; Biswas *et al.* 2004; Fayolle *et al.* 2006). C^1 continuity might also be useful, if the user desires smooth blends, metamorphosis and other operations as function C^1 discontinuities can cause unexpected creases in the resulting surface (Fayolle 2006), as shown in figure 3.1. Therefore, there is a need for a C^1 continuous function ϕ, with values close to the exact distances and exact boundaries to maintain the correct object definition. This problem is tackled in section 3.2.

Transfinite interpolation for heterogeneous volumetric modelling is already a powerful technique to create gradient volumetric properties in an object. It uses distances and distance approximations to produce an interpolation between *material features* with known material attribute values. These material features are defined by the user, and it can be any kind of feature, as soon as its geometry is defined. However, transfinite interpolation does not take into account the shape of the object nor the

relative positions of the features. These issues are tackled in 3.3. Distances in this case are used at several levels to interpolate the volumetric attributes in respect to the shape geometry G.

In chapter 2, the survey on metamorphosis showed that the geometric side of this operation was well understood and studied. However, the metamorphosis changes the volumetric material properties through time. This issue is a good example of an attribute $S_i(f(\phi), t)$ which depends on time. There are only a few available solutions to interpolate between two volumetric material distributions, and most of these involve solving partial differential equations or tracking particles. This problem is tackled in section 3.4, where distances are used to evaluate the material distribution at a time t.

Finally, an example of attribute values used to support complex parametrized operations is given in section 3.5. Several weighting attribute functions S_i are created to mix and deform various pieces of several objects.

The main contributions of this thesis in the area of heterogeneous object modelling are:

- A general formulation of a distance-based time variant heterogeneous object model and the formulation of several specific user-oriented operations within this model
- An approach to providing shape preserving, C^1 continuous field except at the boundaries, with convolution filtering of signed distance fields in section 3.2
- A shape aware *feature-based* interpolation method for complex objects: shape conformal volumetric interpolation in section 3.3
- A shape aware time-dependent volumetric interpolation of two material distributions: space time transfinite interpolation in section 3.4
- A method to mix several objects together selectively based on features: morphological shape generation in section 3.5

3.2 Convolution filtering

In chapter 2, the importance of C^1 continuity for defining functions of FRep objects was established. It is also important to remember that such continuity is desirable for heterogeneous volumetric modelling and gradient material properties. At the same time, exact boundaries need to be preserved to comply with the user's input. In this section, a novel solution which relies solely on a distance function to produce an approximate C^1 continuous function, for points away from the surface, which closely approximates the distance function is presented.

3.2.1 Problem description

Distance fields are important in heterogeneous volume modelling as they can be used to represent both the object geometry and the material features. As shown above, distances are also important in feature-based heterogeneous volume modelling to have predictable transitions between various materials. Distance fields prevent unexpected results in volumetric property interpolation, or when some operations are applied to the field (such as offsets and blends). However, distance fields have C^1 discontinuities, which can cause abrupt transitions when interpolating volumetric properties causing stress concentrations (Biswas *et al.* 2004) or unexpected creases on blends (Fayolle 2006). For the given mesh or other surface model, we are looking for a function with the following properties:

- represents the surface exactly to accurately define an object, including its sharp features
- is as smooth as possible (except at the surface) to allow for smooth transitions and avoid stress concentrations
- remains close to the distance function in order to keep predictability

Additionally, the evaluation of this function needs to be relatively fast, since typically several millions samples are evaluated per model, where

the models can have several hundred thousand polygons.

3.2.2 Criteria of evaluation

To evaluate the quality of a function for this problem, the function needs to satisfy the following criteria:

- Capability of representing the given object accurately
- Smoothness except at the surface
- Close approximation of the distance function
- Computationally fast evaluation at the given point

The smoothness can be evaluated visually using volume slices of the function. Additionally, the selected functions can be compared in real applications to judge the quality of the surface (for shape operations) or volumetric property transitions (for property operations). The smoothness can also be evaluated using edge filtering on the gradient field in three dimensions on a volume slice. A successful method should reduce the visible edges away from the surface.

3.2.3 Existing approaches

There are several ways of representing a mesh with a scalar function. Signed distances (Payne and Toga 1992) are exact, but are not smooth. BSP-fields (Fryazinov *et al.* 2011) and L_p distances (Belyaev *et al.* 2012) are able to represent the mesh precisely, and both have at least C^1 continuity everywhere except at the surface. However, BSP-fields cannot guarantee distance properties, unless min-max operators are used, in which case C^1 discontinuities appear. L_p distances can have distance properties and the deviation can be controlled through the parameter p. However, a single evaluation of the function is of linear complexity with respect to the number of polygons. Therefore, L_p are not practical for meshes with a high polygon count. The evaluation of BSP-fields have a linear complexity at best, making it impractical as well. Finally, Radial

Basis Function (Savchenko *et al.* 1995; Carr *et al.* 2001; Yngve and Turk 2002), Multi-level Partition of Unity (Ohtake *et al.* 2003) and Compact Support Radial Basis Function (Morse *et al.* 2001) only approximate the shape. Additionally, there are no guarantees of the distance properties of those functions. Voxels have been used to calculate the distances, and then smoothing and interpolation is used. While it provides the best performance per sample, and produces a C^1 continuous function (given an appropriate interpolation scheme), it cannot represent the objects accurately. A more complete review is given in section 2.3.

3.2.4 Overview

The presented technique applies convolution filtering to the initial distance field, where the filtering region of $\mathbf{p}$ is reduced to a point if $\mathbf{p}$ belongs to the surface. The filtering region increases as the distance to the surface increases.

The exact signed distance field function f of the object is replaced by the function g:

$$g(\mathbf{p}) = \frac{\int_{\mathbb{R}^3} f(\mathbf{p} - \mathbf{s}\, h(\mathbf{p}))\, w(\mathbf{s}) d\mathbf{s}}{\int_{\mathbb{R}^3} w(\mathbf{s}) d\mathbf{s}} \tag{3.4}$$

where

- w is a smooth kernel function,
- h is a function that controls the kernel size, such that $h(\mathbf{p}) = 0$ when $f(\mathbf{p}) = 0$,
- $\mathbf{s}$ is a displacement vector over the whole space where the signed distance field f is defined.

The function h is discussed in details in subsection 3.2.5, and the function w is explained in subsection 3.2.6.

The scalar field defined by the function g preserves the same zero level set as the function f since $h(\mathbf{p}) = 0$ when $f(\mathbf{p}) = 0$. Thus, $\mathbf{p} - \mathbf{s}\, h(\mathbf{p}) = \mathbf{p}$ and so $f(\mathbf{p} - \mathbf{s}\, h(\mathbf{p})) = 0$ if $f(\mathbf{p}) = 0$.

If $\mathbf{s}h(\mathbf{p})$ is substituted by $\mathbf{u}$, then $d\mathbf{u} = h(\mathbf{p})d\mathbf{s}$ and, providing that

$h(\mathbf{p}) \neq 0$, the parameterized family of functions $w_h(\mathbf{u}) = \frac{1}{h(\mathbf{p})} w(\frac{\mathbf{u}}{h(\mathbf{p})})$ is introduced. Equation 3.4 can be rewritten as follows:

$$g(\mathbf{p}) = \frac{\int_{\mathbb{R}^3} f(\mathbf{p} - \mathbf{u})\, w_h(\mathbf{u}) d\mathbf{u}}{\int_{\mathbb{R}^3} w_h(\mathbf{u}) d\mathbf{u}} \tag{3.5}$$

In order to maintain the same surface, $g(\mathbf{p}) = f(\mathbf{p}) = 0$ needs to be true for all $\mathbf{p}$ lying on the surface of the object.

From the equation 3.5, it is clear that g is a convolution product if the function w_h is not a function of $\mathbf{p}$. In subsection 3.2.5, we show that past a user-defined value, $h(\mathbf{p})$ is constant. Therefore, g is a convolution of f and w_h, when $h(\mathbf{p})$ is constant. The properties of the convolution integral defined by the equation 3.4 heavily depend on the definition of the functions w and h, which are discussed below.

3.2.5 Kernel size function

The function h is introduced to smoothly interpolate values from 0 to 1 as the distance to the surface increases. This function should be continuous on $\mathbb{R}^+$ and such that $h(0) = 0$ and $h(\mathbf{p_t}) = 1$, $\forall \mathbf{p_t} | f(\mathbf{p_t}) \geq f_c$, for a given capping value f_c.

There are three reasons to require that $h(\mathbf{p_t})$ remains constant for points beyond the capping value f_c:

- Only if w_h is not a function of the point p the continuity is guaranteed. If the kernel size is a function of the distance, then a discontinuity is introduced where the distance function is discontinuous.
- The larger the kernel size is, the larger the difference between g and f is. Capping the kernel size means the difference between g and the distance can be bounded.
- If the maximum kernel size is kept small enough, packet sampling can be employed to improve the sampling performance for the numerical evaluation of g.

One way to define such a function h is to use the smooth step function defined by the GLSL Language Specification Version 1.40 (Kessenich *et al.* 2009) as:

$$h(\mathbf{p}) = 3r(\mathbf{p})^2 - 2r(\mathbf{p})^3 \tag{3.6}$$

where $r(\mathbf{p}) = min(\frac{|f(\mathbf{p})|}{f_c}, 1)$.

Note that this function does not depend on the sign of the scalar field, as the unsigned distance value is used. Reducing the kernel size to zero on the mesh surface provides the exact surface representation by the convolution filter of the distance field.

3.2.6 Kernel function

The kernel function w should take its maximum at 0 and smoothly converge to 0 as its argument vector goes to infinity with any of its coordinates. Two obvious ways to define such a function is the Gaussian distribution and a bump function. Two functions are illustrated and discussed below in the one dimensional case, although the formulation is also suitable for multidimensional vectors.

The Gaussian distribution is defined as $w_a(\mathbf{u}) = \left(\frac{a}{\pi}\right)^{\frac{3}{2}} e^{-a||\mathbf{u}||^2}$, where the parameter $a \in \mathbb{R}^+$ controls the width of the Gaussian. The larger a is, the closer g will approximate f. Gaussian curves for different values of a are illustrated in figure 3.2.

The bump function used in the rest of this thesis is defined as:

$$w_b(\mathbf{u}) = \begin{cases} b^{-3} e^{\frac{1}{||\frac{\mathbf{u}}{b}||^2 - 1}} & \text{if } ||\frac{\mathbf{u}}{b}|| < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

where the parameter b controls the width of the bump function. The smaller b is, the closer g will approximate f. Bump functions for various values of b (0.15 in green, 0.2 in yellow, 0.5 in pink and 1 in blue) are illustrated in figure 3.3. The bump function has the advantage of having

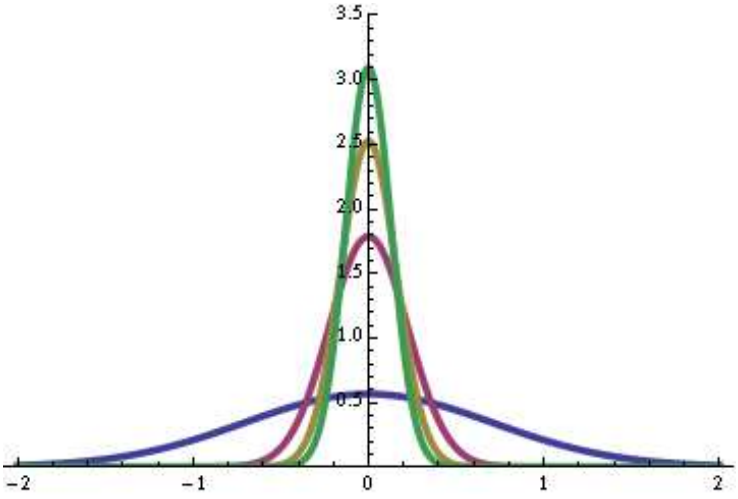


Figure 3.2: *Gaussian weight function for various values of the parameter a*

compact support, unlike the Gaussian curve. This will be useful for its implementation.

Regarding continuity, there are two cases for g:

- $|f| > f_c$: w_h behaves like w (i.e., w_h does not rely on f); so for the weighting functions proposed above g is smooth;
- $|f| \leq f_c$: w_h has discontinuous partial derivatives whenever $|f| \leq f_c$ and f has discontinuous partial derivatives, g is not guaranteed to have derivative continuity.

3.2.7 Selection of parameters

The parameter f_c controls the distance to the boundary of the object represented by f outside of which g is guaranteed to be smooth. This parameter needs to be small enough to exclude points where f is not smooth, or set as small as possible otherwise. f_c defines an envelope around the surface of the shape defined by f, in which the kernel size is not constant. While the kernel size is not constant, the function g is only guaranteed to be C^0 continuous.

The parameters a and b related to how the weight functions control

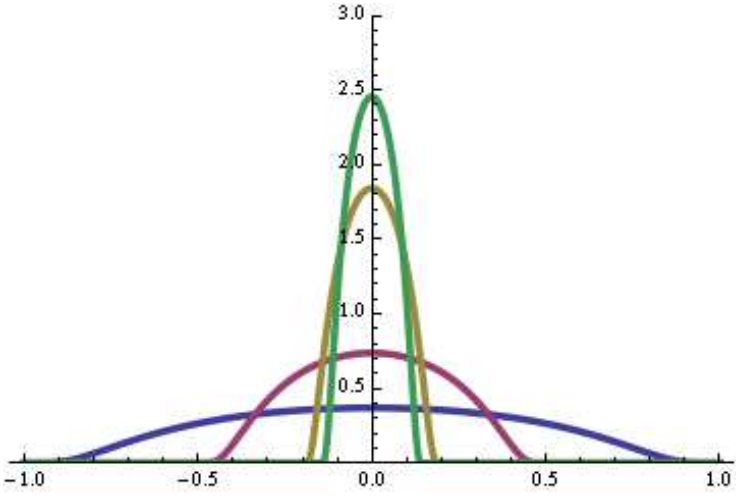


Figure 3.3: *Bump function for various values of the parameter b.*

the width of the Gaussian or bump function. There is a trade off between how close g is to the distance function and the shape of the level sets of g. It should be selected based on the type of applications. If the properties being modelled are closely related to the distance function, then the parameters must be selected to keep g close to the distance function. However, if smoothness is more important and slow transitions are desired, a should be small and b large (e.g., $a = 1$ and $b = 1$).

Figure 3.4 shows the signed distance to the boundary of the interval $[0, 1]$. The Gaussian function is used for w_h with $a = 50$. First, f_c is set to one. Since the discontinuity of the signed distance function happens at 0.5, and $0.5 \leq f_c$, the derivatives will still be discontinuous. Figure 3.5 shows the plot of $g(x)$ corresponding to this case. The derivative discontinuity is visible in figure 3.5b at 0.5.

However, when $f_c = 0.25$, g has derivative continuity (figure 3.6). For this simplistic case, the derivatives of g can be evaluated analytically at $x = 0.5$, and it can be shown that $g'(x) = 0.0$.

While g is continuous, it is possible to create extra zero level sets for some selections of f_c and a (or b). This occurs if the width is large and the capping distance f_c is small. The following subsection discusses the necessary restrictions on the parameters. Figure 3.7 demonstrates this behaviour. The width is set to $a = 2.0$ and the capping distance is set

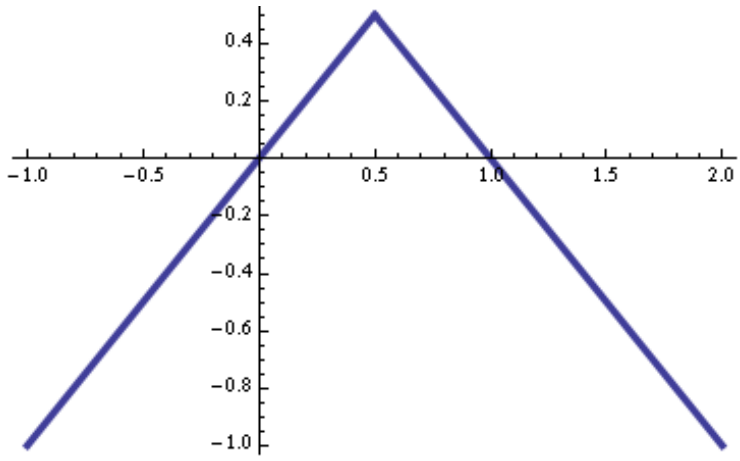


Figure 3.4: *Plot of the distance to the segment* $[0, 1]$.

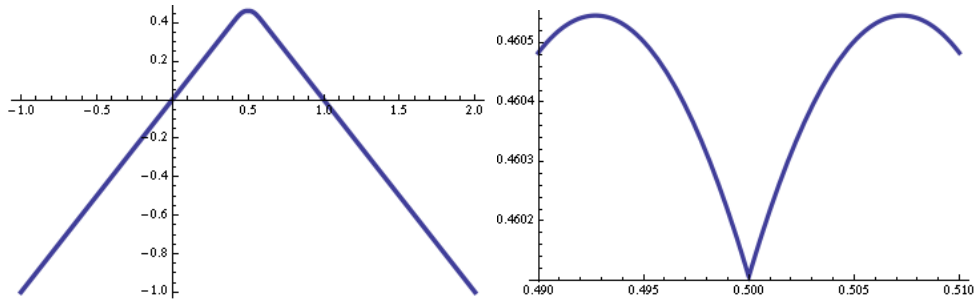


Figure 3.5: *Plot of* g *using* $f_c = 1.0$. *Right: zoom near the point* $x = 0.5$.

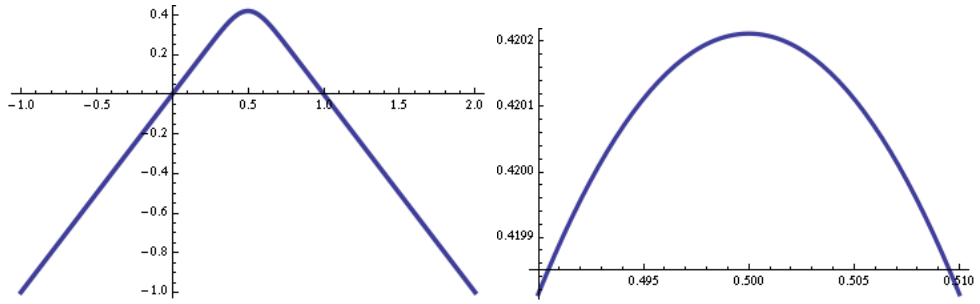


Figure 3.6: *Plot of* g *using* $f_c = 0.25$. *Right: zoom near the point* $x = 0.5$.

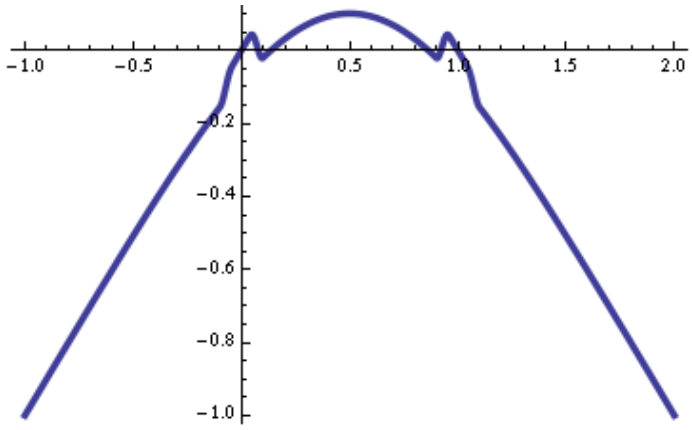


Figure 3.7: *By picking a large width for the Gaussian and a small f_c, it is possible to obtain unwanted extra zero level set.*

to $f_c = 0.1$. As explained above, the width needs to be reduced or the capping distance increased to fix this issue.

3.2.8 Parameter restrictions

The Gaussian function does not have local support which means the parameters have to be selected carefully as described above. However, if a bump function is used, a relationship between f_c and b can be established. In order to prevent additional zero level sets, the filtering region must not cross the surface of the object otherwise the sign of f changes. Therefore, if b is set, then f_c has a lower bound depending on the value of b. Similarly, a given value of f_c implies an upper bound on b. In order to avoid any non-zero weighted $\mathbf{s}$ to cross the surface, the following inequality must hold for all $\mathbf{p}$:

$$b\,h(\mathbf{p}) \leq |f(\mathbf{p})|$$

using equation 3.6, this inequality expands to:

$$b\,(3(min(\frac{|f(\mathbf{p})|}{f_c}, 1))^2 - 2(min(\frac{|f(\mathbf{p})|}{f_c}, 1))^3) \leq |f(\mathbf{p})|$$

$min(\frac{|f(\mathbf{p})|}{f_c}, 1)$ is substituted by X which leads to the following inequal-

ity:

$$b(3X^2 - 2X^3) - Xf_c \leq 0 \tag{3.8}$$

The highest point which could potentially cross $h(x)$ is identified by finding the maximum for X of the above function in the range $[0, 1]$. To do so, the roots of derived function have to be solved:

$$s'(X) = -6RX^2 + 6RX - f_c \tag{3.9}$$

After solving and substitution, in the original inequality, the following relationship between b and f_c is reached:

$$\begin{aligned} b &\leq \frac{8f_c}{9} \\ f_c &\geq \frac{9b}{8} \end{aligned} \tag{3.10}$$

These parameter restrictions will guarantee that no additional zero level set is introduced.

3.2.9 Numerical evaluation of the convolution integral

In this section, a simple method for evaluating the convolution integral of equation 3.4 is shown, based on importance sampling and Monte Carlo integration as shown in Hastings (1970) and further explained in Pharr and Humphreys (2004).

In the general case, the integral defined in equation 3.4 cannot be evaluated analytically and therefore a numerical approximation is required. A discrete filter is applied to the scalar field to make it smooth. In order to evaluate the convolution from equation 3.4, the following finite

summation is used:

$$g(\mathbf{p}) \approx \sum_{i=1}^{n} f(\mathbf{p} - \mathbf{s}_i \cdot h(\mathbf{p})) \cdot w_i \tag{3.11}$$

where n is the number of samples, $f(\mathbf{p})$ is the distance function being filtered at the point $\mathbf{p}$, $\mathbf{s}_i$ is the i-th sample position in a sampling volume, $w_i = w(\mathbf{s}_i)$ is the weight associated to the sample position $\mathbf{s}_i$ and h is the function controlling the size of the kernel as discussed in section 3.2.

Equation 3.11 can be used when the samples $\mathbf{s}_i$ are on a regular grid (or when they are sampled from a unit uniform distribution). This is the most naive implementation of the Monte Carlo integration. It is possible to get better results by sampling, for example, from the distribution with density w (or $\frac{w}{\int w}$ if w is not normalized). In this case, an approximation of equation 3.4 is obtained by:

$$g(\mathbf{p}) \approx \frac{1}{n} \sum f(\mathbf{p} - \mathbf{s}_i \cdot h(\mathbf{p})) \tag{3.12}$$

where $\mathbf{s}_i$ are sampled from the distribution with the density w. This corresponds to the standard Monte-Carlo approximation of integrals with importance sampling (Hastings 1970).

The quality of the result depends on the number of samples, their distribution and weights. If w is chosen to have compact support, and the kernel size is small enough, then the samples will be located in small region of space around the point being evaluated. If the samples are in the same neighbourhood, then packet sampling can be used to evaluate efficiently all the samples at once.

3.2.9.1 Sample distribution and weights

To accurately and efficiently evaluate the convolution integral numerically the samples should be distributed inside some volume. Equation 3.4 suggests sampling in the entire space, at least for the Gaussian kernel, while for the bump function sampling a finite volume is sufficient. For practical reasons, when equation 3.11 is used for the approximation,

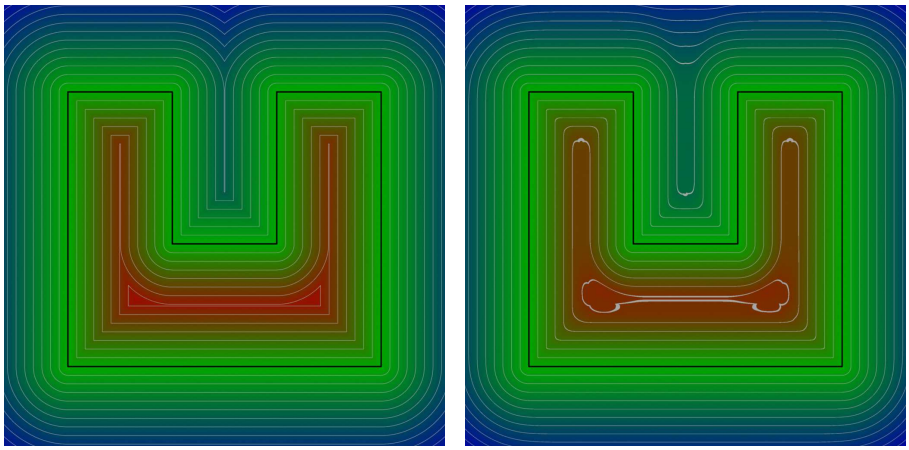
a finite volume is used (called a unit volume) near the evaluation point. The choice of this volume is made by bounding a volume which bounds all the non negligible weights. Two obvious ways to define a unit volume are: a unit sphere centred at the query point and a unit cube. Therefore all the samples are distributed inside a unit cube and defined by their position and weight. As the number of samples increases, the closer the approximation to the integral becomes, but at the same time the less efficient the method becomes. So a balance has to be found.

To distribute the samples inside the unit cube, different approaches can be used. The most naive solution is a regular pattern of a 3^3 lattice and each sample point has a weight given by the function w. However, the resulting field has many visible creases and discontinuities which remain even if the number of samples is increased in a regular grid. Instead, a non uniform sample distribution following the function w is employed. This is the importance sampling method. In this case, more samples are drawn where the weights are larger.

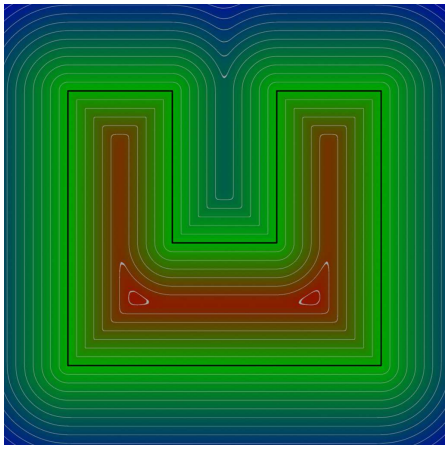
To draw a sample set of N points where N is variable, and the distribution of the points follows the density function w, we rely on adaptive rejection sampling (Gilks and Wild 1992). The sample set is drawn as follows:

1. A sample points $\mathbf{p_i}$ is generated in a unit cube using a uniform distribution.
2. The weight value w_i of the sample point $\mathbf{p_i}$ is calculated.
3. A random number x_i is generated in the range $[min(w), max(w)]$.
4. The sample point is rejected if $x_i < w_i$. When a sample is rejected, the process is repeated from step 1 until it passes the test.

The resulting fields can be seen in figure 3.8. The regular uniform filter has visible C^1 discontinuities just like the signed distance field. The Gaussian distributed samples provide smooth fields. In practice, numerical filters provide good results while preserving the efficiency of the method. Figure 3.9 shows how the number of samples affects the quality of the field.

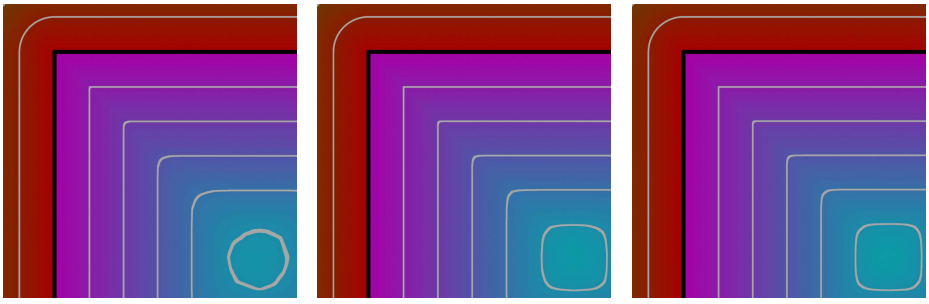


(a) (b)



(c)

Figure 3.8: *Filtering the distance field: (a) shows the signed distance field, (b) uses a regular grid with uniform weights, (c) use a random Gaussian distribution of samples with Gaussian weights.*



(a) (b) (c)

Figure 3.9: *Filtering of distance fields with (a) 32 samples, (b) 64 samples, (c) 92 samples.*

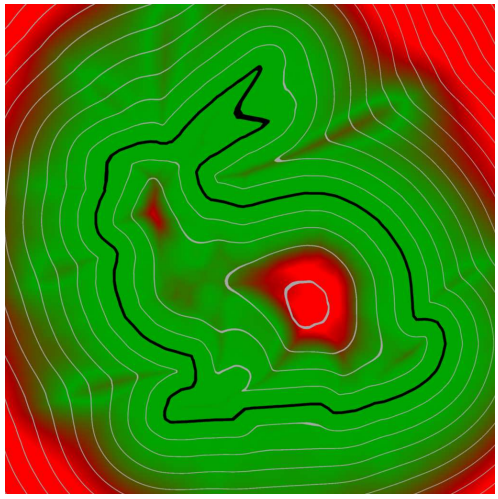


Figure 3.10: *Medial axis detection using the initial samples. Green indicates low sampling rate while red means high sampling rate. When the field is red, we are close to a discontinuity, and therefore require high sampling rate. A small number of samples in the green areas are sufficient to get a good approximation.*

This method could possibly benefit from low discrepancy sequences instead of the lattice used in figure 3.8b. Quasi-Monte Carlo integration uses (Morokoff and Caflisch 1995) uniformly distributed sequences which improve the convergence. Caflisch (1998) compares Monte Carlo with importance sampling and quasi Monte Carlo integration, and concludes that some techniques can be used to make quasi Monte Carlo integration perform at least as well as importance sampling.

3.2.9.2 Adaptive quality

As mentioned before, the number of samples has an impact on the quality of the result and the efficiency. To achieve good quality without sacrificing efficiency in the computations, the number of samples is adaptively changed across space. A low number of samples is able to approximate the convolution reasonably well when far from the surface or the medial axis. Therefore, the number of samples should be adaptively increased around the surface and the medial axis.

Finding the distance to the medial axis is not a trivial task, as the medial axis of the distance field is not defined explicitly. However it can be

approximately determined by analysing the difference between the value in the central point of the space and the values of the neighbourhood of this point as observed in Gagvani and Silver (1999). The farther the query point from the medial axis, the closer the neighbourhood average to the central point value will be. Figure 3.10 shows the number of samples (green for N_{min}, red for N_{max}) when using the medial axis detection field as introduced in Gagvani and Silver (1999). If the first samples are the uniform $3 \times 3 \times 3$ lattice, built around the sampling point, and as wide as the kernel size, then they can be used to build a neighbourhood average. Finally, a smooth step is used as a transfer function to control the number of samples based on the distance. Both of those functions are mixed together and interpolate between the minimum and maximum number of samples.

$$n_a(\mathbf{p}) = N_{min} + \frac{n_{medial}+n_{distance}}{2} \cdot (N_{max} - N_{min})$$

The function n_a provides a real value to control the number of samples. Note that using just the integer part of this value is not effective because the change in the number of samples introduces noise in the resulting field because of the discrete jumps. Instead, the fractional part is used to weight the last sample in the summation.

3.2.10 Summary

In this section, a method for generating a scalar field which is close to the exact distance and C^1 continuous everywhere except at the surface was introduced. The convolution filtering method provides good user control on various demands on smoothness. The parameters adjustments and dependencies were explained to show the impact that such parameter values have on the field as well as the limitations and constraints that parameters have on each other. The main advantages of convolution filtering are:

- Any distance field can be used regardless of its source;
- The smoothness of the field is controllable;

- It preserves the input zero level set.

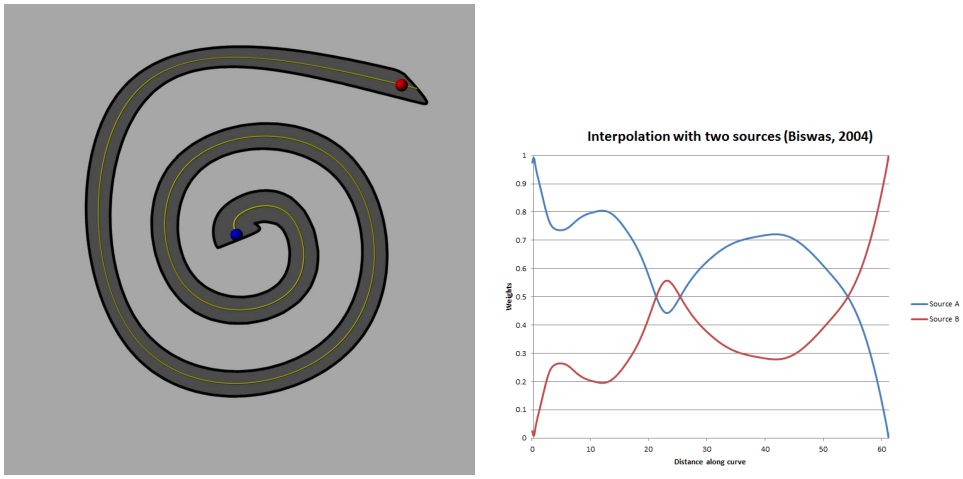
Shape preserving smooth distance fields introduced in this section will be preferred in section 3.4 and 3.5 to the exact distance fields. The evaluation, applications and results of convolution filtering are shown in section 4.2. The work on convolution filtering presented in this section was published in Sanchez *et al.* (2015).

3.3 Shape Conformal Volumetric Interpolation

In this section, a novel method for shape conformal volumetric interpolation to define functionally graded material is presented. In section 2.2, several limitations of the transfinite interpolation method presented in Biswas *et al.* (2004) were shown. Biswas *et al.* (2004) can produce counter-intuitive results which can be attributed to two reasons. First, concave shapes lead to non-intuitive results because the interpolation scheme relies on Euclidean distances. For the results to reflect the user's intention, the interpolation needs to be shape aware, which means the distance measure needs to reflect the distance within the shape. Additionally, to reflect the user's intent, the accessibility of the points in respect to the various material features have to be considered.

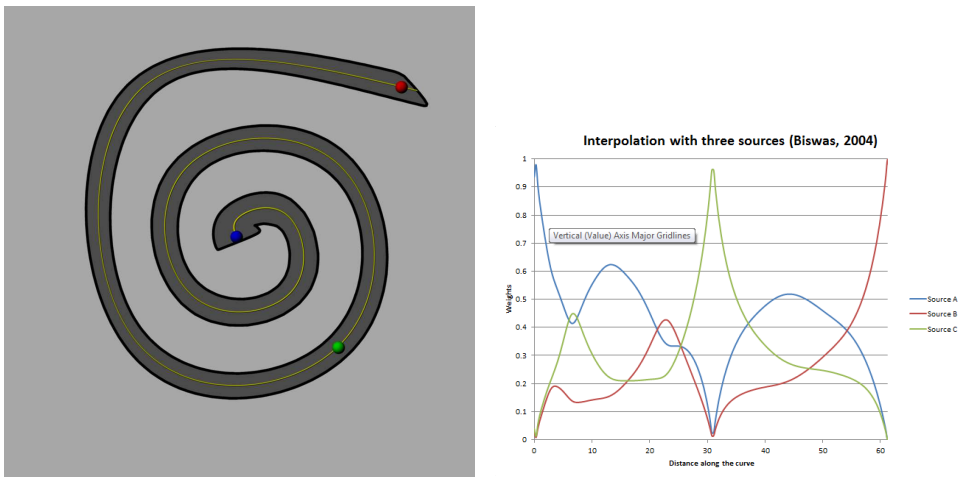
3.3.1 Problem description

As mentioned earlier, *feature* based heterogeneous volume modelling is a powerful concept to define gradient material properties. The user defines non overlapping features (point, line, curve, volume) with a known homogeneous properties. In areas where no feature has been defined (*material gaps*), the properties of the features are interpolated smoothly. This is an important technique because it allows non-technical users to easily define gradient material properties. Biswas *et al.* (2004) uses the transfinite interpolation to interpolate material properties between a number



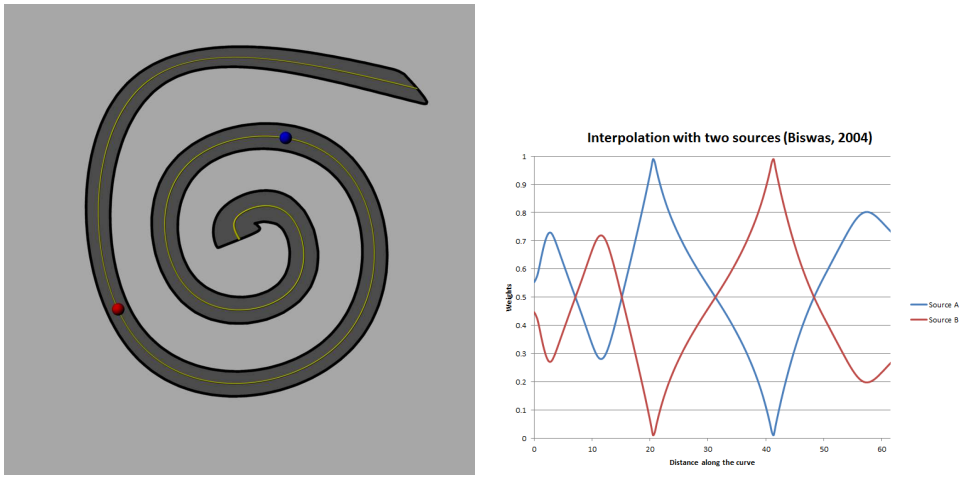
(a) (b)

Figure 3.11: *(a) A simple spiral, and two sources: one blue at the centre, and one red at the end. (b) The weights generated by Biswas* et al. *(2004) with Euclidean distances.*



(a) (b)

Figure 3.12: *(a) A simple spiral, and three sources: one blue at the centre, one green in the middle and one red at the end. (b) The weights generated by Biswas* et al. *(2004) with Euclidean distances.*



(a) (b)

Figure 3.13: *(a) A simple spiral, and two sources: one third along the curve, and one red at two third of the curve. (b) The weights generated by Biswas* et al. *(2004) with Euclidean distances.*

of features. However, this method does not take the shape of the object into account and the relative positions of features within the shape. This leads to counter-intuitive results. For instance, for a simple spiral and some material sources (see figures 3.11a, 3.12a, 3.13a), the weights distribution along the medial axis of the spiral shape do not seem intuitive (see figures 3.11b, 3.12b, 3.13b).

The two identified limitations of this method are:

- The method does not take into account the shape, and solely relies on Euclidean distances to features, see figure 3.11b or figure 3.12b
- The method interpolates attributes at any point in the material gaps. However the user may want to localize the interpolation of the properties, and keep a sense of occlusion between the features. See figure 3.13b for instance.

3.3.2 Criteria of evaluation

To evaluate the proposed method, we will compare it with the work of Biswas *et al.* (2004) for several simple shapes showcasing the issues mentioned above to confirm the differences between a shape conforming

interpolation and the traditional distance-based interpolation. For each shape, the visual results can be compared, however it can also be evaluated by plotting a graph for some of the simple shapes. For a spiral in 2D or a spring in 3D, samples distributed along the centreline can provide a good estimate of the expected results or highlight the issues. Monotone behaviour of the obtained function along such a curve means the interpolation is shape conforming, while some oscillations mean that the interpolation is shape independent. For some shapes, a single curve is not sufficient. In such cases, we can draw a number of curves along the medial axis or parts of the skeleton.

We will introduce curves for each of object and analyse the weights distribution along this curve produced by our method and by the transfinite interpolation with Euclidean distances.

3.3.3 Existing approaches

In section 2.2, several heterogeneous object modelling methods were shown. In this section, we are interested solely in non-evaluated, scalar field based methods. The work of Siu and Tan (2002b) introduced the concept of source-based heterogeneous volumetric modelling. The user can define several types of features (points, lines, planes). Those features can then be used to blend materials based on the distance to those features. A function maps the distance to the feature to a material value. The work was extended to multiple features by using a weighted sum. This solution has several limitations. First, it assumes the object is made of a basic material, and then is enhanced through some features. All the parameters have to be adjusted to the size of the object, and the interpolation is local to the feature. Additionally, it relies on the Euclidean distance, which does not take the shape into account. Biswas *et al.* (2004) proposed a global interpolation, where the user defines features (point, lines, solids) of known homogeneous properties, and then any point outside the material features interpolates all of the feature properties smoothly. It is a significant improvement over Siu and Tan (2002b) since it requires less parameters, and interpolates

the values at any point in the shape, not just within a radius. However, this solution is independent of the object geometry. Additionally, the global interpolation means that points far from all features take almost equal weights from each feature property, regardless of the shape of the object. Khoda *et al.* (2013) acknowledged the issue, and used the medial axis to provide a shape-aware internal structure. However, the medial axis is often problematic to obtain for arbitrary models. Finally, to define the features, all these techniques rely on Euclidean distances. If the analytical function is used, then C^1 discontinuities appear, and can cause visible creases in the material transitions. Fayolle *et al.* (2006) provided smooth approximations of min/max (Ricci 1973) operators to construct features and maintain an approximate signed distance. However, the users need to construct features using Boolean operators on the provided primitive shapes.

In conclusion, no existing method addresses shape conforming interpolation issues outlined above.

3.3.4 Overview

Shape conformal volumetric interpolation method relies on *feature based* modelling, as described in section 2.2.5. First, interior distances to material *features* are computed. The details are presented in subsection 3.3.5. We may want to partition the object to localize the influence of each source in some cases. A Voronoi diagram is built using these distance fields. The Voronoi sites are the *features* defined by the user. These features can be any geometrical element from which an interior distance can be calculated (such as points, curves, meshes). The details are presented in subsection 3.3.6. Various offsets are applied to each Voronoi cell to move the cell boundaries to encompass the original feature, but create adjustable material gaps where the interpolation is performed, which is described in subsection 3.3.7.

Figure 3.14 shows the steps of our method. Four material features are defined in an object (see figure 3.14e), and the interior distance field

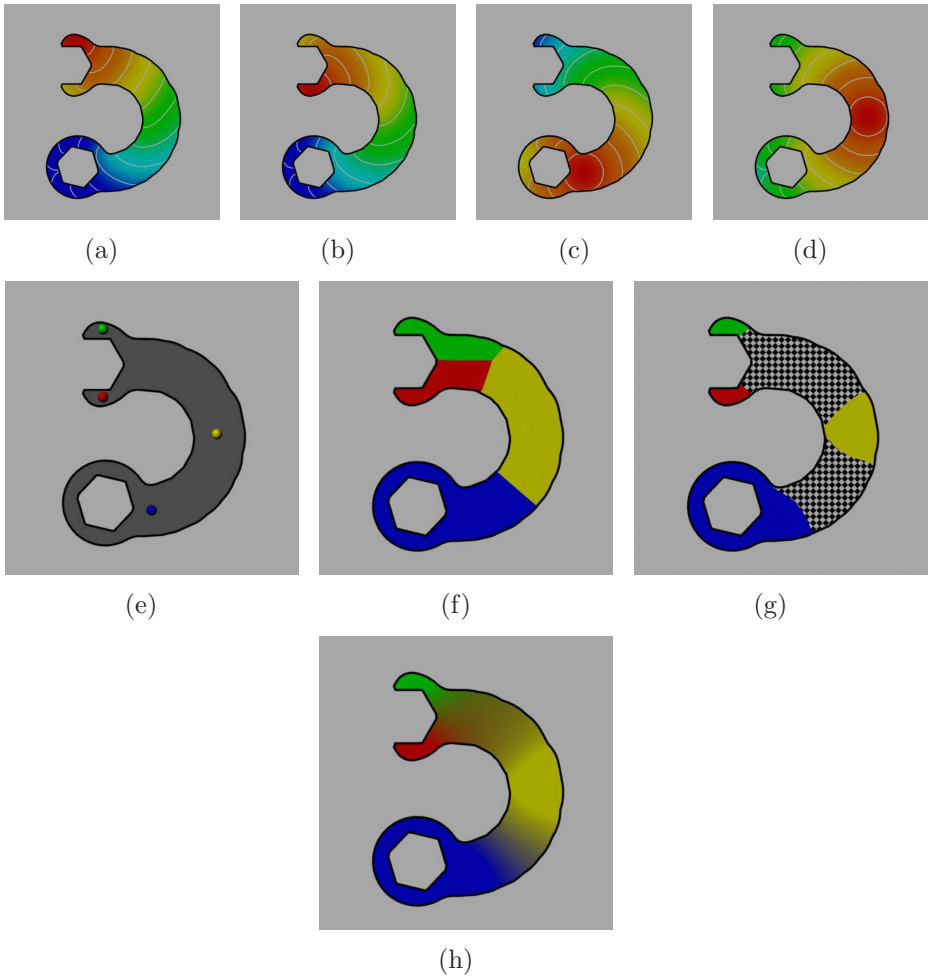


Figure 3.14: *Overview of the steps of our method: a,b,c and d) The interior distance fields for each material feature are computed, e) The Voronoi diagram is built based on those distance fields, f) The Voronoi cells are shrunk to make gaps where the interpolation should happen, g) The result of our interpolation*

to each material feature is calculated (figure 3.14a, 3.14b, 3.14c, 3.14d). Next, these interior distance fields are used to create a Voronoi diagram (figure 3.14f). Then, the cells of the diagram are shrunk to create a material gap. The checker pattern in figure 3.14g show the material gap. Finally, the interpolation is performed in the material gap (figure 3.14h).

For the following sections, the following notations will be used:

- f_s is the function of the shape representing the object
- d_i is the interior distance function of the *material feature* i within the shape represented by f_s
- n is the number of *material features*

3.3.5 Interior distance field for material features

The exact interior distance is a NP-hard problem (Canny and Reif 1987), and as stated in section 2.3.1.3, various approximation of the interior distance have been proposed for meshes and other representations. For FRep objects, there are not many choices. The approximation of the interior distances can be computed on a voxel grid with a smooth interpolation in between the voxels.

To calculate the interior distance, Fast Marching Methods (FMM) can be used as shown in Adalsteinsson and Sethian (1995). FMM are fast and accurate. The only limitation, shared with any other voxel-based method, is that it may not capture all the details of the source feature. Alternative solution are provided in section 2.3.1.3. Additionally, we provide a simple algorithm to calculate an approximation of the interior distance based on Djikstra's algorithm in appendix A, algorithm 1.

Interior distances are only defined within the shape. However, there are some benefits to being able to provide a value at any point in space for technical reasons.

- If a shape is made of several disconnected components, for any point in the second component, interior distances are not defined.

- The interior distances are calculated on voxel grids. It means that thin features may not be captured. A point in a thin feature could be missed. This is true if a thin feature connects two parts of the same object, or if two thin features come close but do not connect.
- The interior distance is retrieved by performing an interpolation of neighbour voxel values, but it is problematic for points near the object surface.

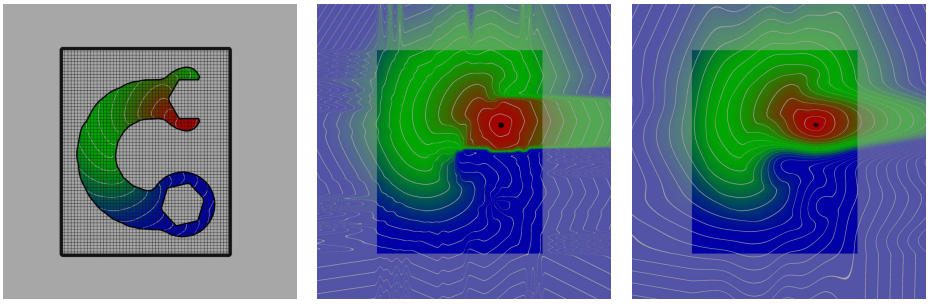
For all these reasons, we need to be able to provide values anywhere in space. Based on these issues, we propose to extrapolate the interior distance field to the outside of the object to overcome these issues.

- When a point is close to the surface, its value should be close to the interior distance at closest point to the surface.
- The distance needs to increase as the sample point is further from the surface, so that we get an approximation of the Euclidean distance if the voxel grid fails to capture parts of the geometry.
- The field should be smooth to avoid unexpected behaviour during interpolation.

Note that if the voxel grid is too coarse, we may misinterpret disconnected parts as connected, or connected parts as disconnected. Since there is no clear solution, we assume that the grid may miss features, but not topologically important features. That is, the rasterization can miss features (such as pins and needles) but not features which connect two parts of a shape together.

The objective here is to give intuitive values outside the object while keeping the discontinuity as far from the surface as possible. Since the values are computed on a lattice, there are two cases:

1. If the point lies inside the bounds of the lattice, a two-step procedure is applied. First, the shortest path to the voxelized shape is calculated, and then the interior distance from the surface point to the material source is added. Secondly, a smoothing pass is applied to the voxels outside the object.



(a) (b) (c)

Figure 3.15: *The interior distance field inside and outside the shape: a) The interior distance field for the shape, and the voxel lattice in black, b) the extrapolation without the last blurring step, c) the extrapolation with blur. The distances are mapped to a colour ramp, where red represents small distances and blue large distances. The white lines are uniformly spaced iso-lines. The faded colours indicate values outside the voxel grid.*

2. If the point lies outside the lattice, values are extrapolated to preserve the rate of change at the boundaries of the lattice

For the first stage, a Euclidean distance transform is used. The values of the samples that lie within the bounds of the lattice but outside the object can be found by using a distance transform. The boundary is provided by the first propagation (O in algorithm 1). Once the distance transform is complete, we carry over the values of the interior distances to the voxels outside the shape. The distance transform keeps the discontinuity as far from the surface as possible. This means that the boundary cells on the outside of the shape have values close to the ones on the inside. This is necessary to prevent the cubic interpolation from creating noise near the surface or on the surface. The result of this stage is shown in figure 3.15b. As shown, the resulting field is not smooth. In the second stage, we apply a blur to the values outside the object, so that the interior distance remain as precise as possible while the extrapolated values are smooth. The resulting field is shown in figure 3.15.

To sample any point in space, an extrapolation is needed. For simplicity, the discrete data is extrapolated on the fly from the boundary values of the lattice. The extrapolation needs to maintain the rate of change the boundary cells have and carry over their value. This simple

formula fulfils all the requirements and provides good visual results:

$$\begin{aligned} V(x,y,z) &= V(x_b,y_b,z_b) \\ &+V(x_d,y_b,z_b) \times d(x,x_b) \\ &+V(x_b,y_d,z_b) \times d(y,y_b) \\ &+V(x_b,y_b,z_d) \times d(z,z_b) \end{aligned} \tag{3.13}$$

where x, y, z are the integer sample coordinates and are not bounded by the grid, x_b, y_b, z_b are the bounded (closest) coordinates of x, y, z within the grid, x_d, y_d, z_d are bounded coordinates offset by one within the grid. This means that if x is lower than x_b, then $x_d = x_b+1$, otherwise if x is greater than x_b then $x_d = x_b - 1$. The function d simply returns the distance between the two coordinates.

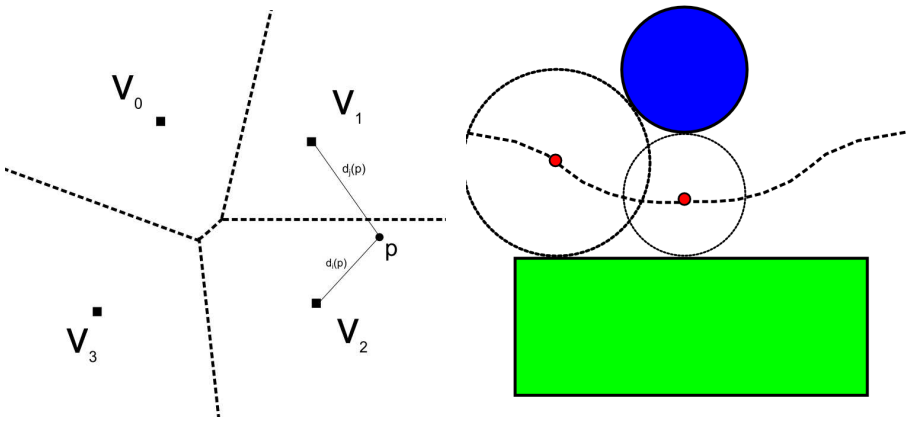
The results of the extrapolation of interior distances are shown in figure 3.15c. The faded colours indicate that the values were outside the voxel grid, and the object is shown in 3.15a for reference. Without the smoothing (figure 3.15b), the field is not smooth. After the smoothing pass, an acceptable field is obtained. The large value gap at the medial axis is as far as possible from the surface, the values are reasonably smooth everywhere outside.

3.3.6 Voronoi diagram using the interior distance

In this method, the Voronoi diagram is create from scalar fields. The following definition is explained with points for simplicity. The minor modification for more complex features will be explained afterwards.

Given a set of n points, which will be called Voronoi sites, $\mathbf{v}_i, i = 1...n$, the Voronoi diagram V is the collection of subsets V_i, where each subset, called Voronoi cells, can be defined by the following Levy and Bonneel (2013):

$$V_i = \{\mathbf{p} \in \mathbb{R}^3 | \, \|\mathbf{p} - \mathbf{v}_i\| \leq \|\mathbf{p} - \mathbf{v}_j\| \, \forall j\} \tag{3.14}$$



(a) (b)

Figure 3.16: *A Voronoi diagram of four sites (a) and of two volumes (b)*

Here $\|\mathbf{p}-\mathbf{v_i}\|$ is the Euclidean distance between the two points. Each pair of Voronoi sites $\mathbf{v_i}$ and $\mathbf{v_j}$ defines a plane that is equidistant from these sites (dashed lines in figure 3.16a):

$$\{\mathbf{p} \in \mathbb{R}^3 | \, \|\mathbf{p} - \mathbf{v}_i\| = \|\mathbf{p} - \mathbf{v}_j\|\} \tag{3.15}$$

The Voronoi diagram can be generalised from points to volumes by replacing the Euclidean distance $\|\mathbf{p}-\mathbf{v_i}\|$ in the equations 3.14 and 3.15 with the value of the distance field for the volume. Consequently, the boundaries between Voronoi cells are not planar any more (red points in figure 3.16b are equidistant from both features, but the boundaries are not straight) and can be curved surfaces defined as:

$$d_i(\mathbf{p}) = d_j(\mathbf{p}) \tag{3.16}$$

d_i and d_j represent the signed distance functions from the point $\mathbf{p}$ to the volumes.

Each surface defined by equation 3.16 is a boundary of a half space which is defined as follows:

$$h_{i,j}(\mathbf{p}) \geq 0 \; \textit{where} \; h_{i,j}(\mathbf{p}) = d_j(\mathbf{p}) - d_i(\mathbf{p}) \tag{3.17}$$

Note that the functions d_j and d_i are not necessarily Euclidean distances. In fact, d_i and d_j are interior distances in our case. The formulations require no modification to support interior distances.

A Voronoi cell i can be defined by the intersection of all the function $h_{i,j}$ for any $i \neq j$.

$$v_i(\mathbf{p}) \geq 0 \text{ where } v_i(\mathbf{p}) = h_{i,1}(\mathbf{p}) \wedge \ldots h_{i,j}(\mathbf{p}) \wedge \cdots \wedge h_{i,n}(\mathbf{p}) \text{ with } j \neq i \tag{3.18}$$

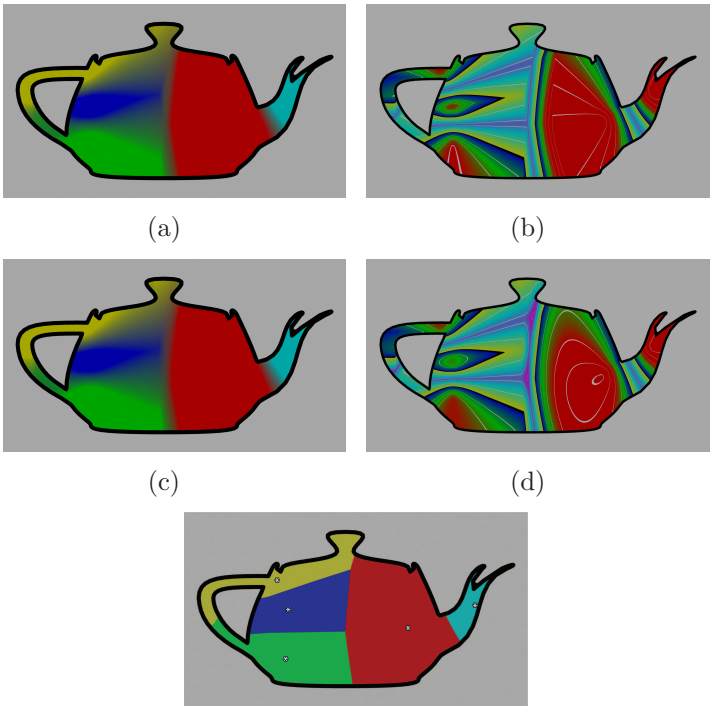
Here $\wedge$ denotes any intersection R-function, however it is preferable to maintain distance properties and have commutative and associative properties. Distance properties are important since we will require the distance to each Voronoi cell in the later stages, and commutative and associative properties are desirable so that the order of operations does not affect the field. Finally, C^1 continuity can be useful to provide smooth transitions.

In section 2.4.1, we showed a comparison of various R-functions. Rvachev (1982); Fayolle *et al.* (2008) fulfil the requirements, however Rvachev (1982)'s simpler implementation makes it a better candidate.

Figure 3.17e shows the teapot model with five material features marked by white stars, each given a constant colour. In this case, the material features are just points, but this method works with any type of feature such as curves, surfaces and volumes. Figure 3.17b and 3.17d show the field for the Voronoi cells after an offset has been applied. The white lines show iso-lines at constant interval, and the colours indicate the values of the field. 3.17b uses the intersection R-function from Ricci (1973) and 3.17b the intersection R-function in Rvachev (1982). The benefits of $\overset{0}{\Re}$ function are the smoothness of the field, and better distance properties for the cell functions.

3.3.7 Material interpolation between features

Since the Voronoi cells use interior distances, those cells are conformal to the shape and do not create disconnected components within the shape



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

Figure 3.17: *Different R-functions influences the field for material interpolation differently: a) Min-max functions (Ricci 1973), b) The field for min-max functions (Ricci 1973), c) R-function* $\overset{0}{\Re}$ *(Rvachev 1982), d) The field for* $\overset{0}{\Re}$ *(Rvachev 1982), e) Voronoi cells for each material feature. Each feature is given a colour.*

(see justification in appendix B). Each cell is a single component solid within the shape which cannot be guaranteed by Voronoi diagrams using Euclidean distances.

However, to perform the interpolation, we need to create a gap between the cells. The cells will have the material properties of their seed (material feature), and the attributes will be interpolated in the gaps in-between the cells. The objective is to interpolate the attributes so that the original material features provided by the user keep their attributes, but the interpolation zone should be as large as possible.

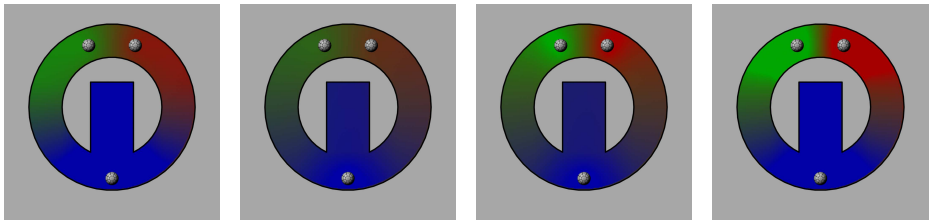
In order to perform the interpolation, two subspaces are needed. The Voronoi cells are adjusted with offsets to shrink the cells. Those shrunk cells are called *feature* cells. The two subspaces are:

- The interpolation zones where the materials from different *feature* cells are interpolated; transfinite interpolation can be applied to the values of the *feature* cell.
- The interior of the *feature* cell for which the material attributes are known.

The interpolation zone can be defined by applying an offsetting operation to the functions $h_{i,j}$ and $h_{j,i}$ in order to create gaps between the cells i and j. The field for the interpolation zone between cells i and j can be defined as follows:

$$b_{i,j}(\mathbf{p}) = -|h_{i,j}(\mathbf{p})| + s_{i,j} = -\left\|d_j(\mathbf{p}) - d_i(\mathbf{p})\right\| + s_{i,j} \tag{3.19}$$

Here $s_{i,j}$ defines the offset value and hence the width of the interpolation zone between two feature materials i and j. Different values for the offset can be chosen but symmetry must be maintained ($s_{i,j} = s_{j,i}$), and $s_{i,j}$ must be positive to avoid overlapping *feature* cells. The first and simplest option is to choose the value of the offset uniformly, i.e. such that $s_{i,j} = s; \forall i, j$, however this might lead to undesirable effects. For example, in figure 3.18a, a small value for the offset was used so that the interpolation would keep the material green and red *features* intact. As



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

Figure 3.18: *Adjusting $s_{i,j}$ offsets to sources: a) Small uniform offset value leads to a large constant blue block at the bottom, b) Large uniform offset value leads to washed out colors for close source features, c) Using an adaptive offset value based on distance between sources leads to intuitive results, d) The adaptive value can be scaled to offset where the interpolation starts.*

a result, the transition between the green and blue material features is short and the blue *feature* seems to expand more than it should. On the other hand, increasing the uniform offset value causes the features near to each other to blur (figure 3.18b). A wiser choice is to adjust the offset for each pair of sources to the relative distance of two material sources, such as the Euclidean or interior distance between these sources. Figure 3.18c shows offset values $s_{i,j}$ adjusted to fit interior distances between features. In figure 3.18d, the transition distances were scaled to give more room for constant properties.

As *feature* cells cannot overlap, and all *feature* cells preserve good distance properties, the general formulation of the transfinite interpolation can be applied. The general inverse distance weighting function described in Biswas *et al.* (2004) using the feature cells as material features is applied to find the weight values w_k. The weighting function is defined as follows:

$$w_k(\mathbf{x}) = \frac{\prod_{m=1;m\neq k}^{n} b_{k,m}(\mathbf{x})}{\sum_{m=1}^{n} \prod_{p=1;p\neq m}^{n} b_{k,p}(\mathbf{x})} \tag{3.20}$$

The final value for the interpolated material is the weighted sum of

the materials of the features for the given point $\mathbf{p}$:

$$\mathbf{c}(\mathbf{p}) = \frac{\sum_{k=1}^{n} w_k(\mathbf{p})\tilde{\mathbf{c}}_k}{\sum_{k=1}^{n} w_k(\mathbf{p})} \tag{3.21}$$

where

- n represents the number of material features for the given point,
- $\tilde{\mathbf{c}}_k$ is the value of the k^{th} material property,
- w_k is the weight for the attribute $\tilde{\mathbf{c}}_k$

It is important to note that equation 3.21 uses a weighted sum. This means that any arbitrary attribute can be interpolated and is not limited to just colours, however, some values do not necessarily behave well with weighted sums. For instance, orientations cannot be interpolated linearly, and weighted sums cannot be used. In such cases, the interpolation scheme should be adjusted according to the attribute type.

3.3.8 Summary

In this section, a new method to interpolate volumetric attributes inside an object was presented. Its objective was to remove two major issues with the transfinite interpolation. This method enables designers to intuitively create gradient materials regardless of the shape or topology of the object. Shape conformal volumetric interpolation also can be adjusted by designers to change the width of the interpolation. It is made up of two parts which can be taken individually depending on cases. For instance, the typical transfinite interpolation can be used with interior distances if features are at corners of the shape, and Euclidean distances can be used if the shape is convex. The evaluation, applications and results are shown in section 4.3. The work presented in this section was published in Fryazinov *et al.* (2015).

3.4 Space time transfinite interpolation

In chapter 2, we showed that transformation of material properties in metamorphosis is problematic with surface based solutions, while volumetric solutions are often discrete and time consuming. The problem is that of interpolating two material distributions, with respect to the shapes associated with each material distribution. In this section, the transfinite interpolation is extended to space time in order to perform this interpolation. Note that the proposed space time transfinite interpolation can be applied to interpolate either the material attribute field values directly or the parameters influencing the fields. This technique can work with any FRep object, but behaves more intuitively with distance fields, and the smoothness of the material properties is improved by the convolution filtering technique introduced in the previous section.

3.4.1 Problem description

Changes of volumetric properties in time occur in various situations such as object deformation, ageing, chemical evolution. Geometric metamorphosis from one volume object to another is an example of a time variant shape transformation that can also involve changes in volumetric material properties. The geometric metamorphosis has been studied in depth, for both BReps and scalar field based methods (Hughes 1992; Cohen-Or *et al.* 1998; Turk and O'Brien 1999a; Galin *et al.* 2000) (refer to section 2.4.2 for more details). However, the transformation of material properties during the metamorphosis received less attention.

The interpolation of volumetric properties through time during metamorphosis should maintain the exact volumetric properties given at the beginning and end of the metamorphosis. For the time in between, the volumetric properties should be smoothly changing from the volumetric properties at the beginning to the volumetric properties at the end of the process. The initial and final shapes should have an impact on the interpolation of the volumetric properties. For instance, the linear interpolation would simply interpolate the material distributions regard-

less of the object shapes and their transformation. This is problematic especially for the points where one of the objects has no defined volumetric properties. This means that features of an object that do not exist in the target object could simply disappear instead of being smoothly transformed. Finally, the transformation function should be evaluated fast, so that a user can get immediate feedback. This also implies that the function for an intermediate object can be sampled at any point in time without requiring heavy numerical computations. In summary, the requirements to this transformation are:

- Interpolation of arbitrary volumetric properties through time
- Interpolation driven by source and target shapes
- Fast queries at any point in space and time
- Easy to use and control
- Capable of interpolating small scale details.

3.4.2 Criteria of evaluation

The evaluation of different approaches can be based on the requirements outlined above. The most critical criteria are the way initial shapes influence the transformation process and the speed of the intermediate values calculation to facilitate real-time applications.

3.4.3 Existing approaches

The most naive solution would be a linear interpolation of the material distributions. The interpolation is independent of the shapes involved. This means that the value of an attribute of a point in space and time relies solely on attribute functions of the objects and the time. Ideally, the value of the attribute would depend on the geometry of the objects.

Currently, there are two approaches for the interpolation of surface properties in time: particle systems methods and PDE-based methods.

Particle systems are used to follow surface properties and flow from the *source* to the *target* object. Drastic changes in topology tend to provide unrealistic or undesirable effects. The work of Smets-Solanes (1996) improved on the above mentioned particle system technique to get good results for animated implicit surfaces. A set of points on the surface of the initial shape (virtual skin), parameterised in 2D, is transformed using different vector fields defining the particle velocities. The work of Tigges and Wyvill (1998) extended texturing through the use of particles to deal with gradient discontinuities that may arise with constructive models and distance functions. Particle system solutions are simple to implement, however, they all share a number of issues. First, they cannot be evaluated independently, and so the results rely on the previous state, and on the number of steps. This means that there is a significant computational cost to rewinding in time, and it also means that the time steps need to be guessed or provided by the user, which will affect the quality of the results. Changing the time step could also drastically change the results too. Additionally, the finer the details are, the more particles are needed. This can be problematic for properties with fine details.

Alternatively, PDE-based methods track surface changes. In Dinh *et al.* (2005) a method was introduced to track surfaces with corresponding attributes in time by solving PDEs. This solution gives consistent results with any topological changes including splits and holes. In the work of Bojsen-Hansen *et al.* (2012), the authors used a similar method to track properties (such as colours and displacement information) during the shape transformation process in morphing or in fluid simulation with heterogeneous fluids. This method is able to handle drastic topology changes. However, solving the PDEs is time consuming and makes this solution impractical for real time applications, in long animation sequences or when applied to large meshes.

Finally, both groups of solutions are limited to surface attributes (usually colours). As such, the interpolation of arbitrary volumetric properties through time remains an open problem. The linear interpolation does not satisfy most of the requirements as discussed above in this sec-

tion.

3.4.4 Overview

In subsection 3.4.5, we will first introduce the concept of space time transfinite interpolation and provide its mathematical formulation, in the simplest case of a single-partition object, with a constant colour or attribute. Afterwards, in subsection 3.4.6, we will extend the formulation to partitioned objects, without established correspondence between the partitions. The method is extended to the case of attributes continuously varying. Finally, better user control is provided through simple parameters in subsection 3.4.7.

3.4.5 Single-partition objects

Let $f_1(\mathbf{p})$ and $f_2(\mathbf{p})$ be defining functions of the initial object G_1 and target object G_2, and c_1 and c_2 be respective colours or other constant material attributes assigned to each of the objects respectively. Here, f_1 and f_2 are signed distance fields or smooth signed approximate distance fields as detailed in section 3.2.

Let $G(t)$ be the shape transformation between objects G_1 and G_2 respectively, and let $\mathbf{c}(t)$ be an attribute transformation between attributes $\mathbf{c}_1$ and $\mathbf{c}_2$ on the interval of time $t \in [0, 1]$ such that $G(0) = G_1$ and $G(1) = G_2$, $\mathbf{c}(0) = \mathbf{c}_1$ for all the points with $f_1 \geq 0$ (on the surface and the interior of G_1) and $\mathbf{c}(1) = \mathbf{c}_2$ for all the points with $f_2 \geq 0$ (on the surface and the interior of G_2). The pair $(G(t), \mathbf{c}(t))$ constitutes a function-based model of a volumetric metamorphosis where both geometry and volumetric properties change in time as two interrelated processes. This formulation leaves the time-variant shape transformation choice up to the user. Numerous scalar-field based metamorphosis are suitable (see section 2.4.2). Shape transformations are used to get an intermediate shape between G_1 and G_2, and any method can be used, but the focus of this work is the function $c(t)$.

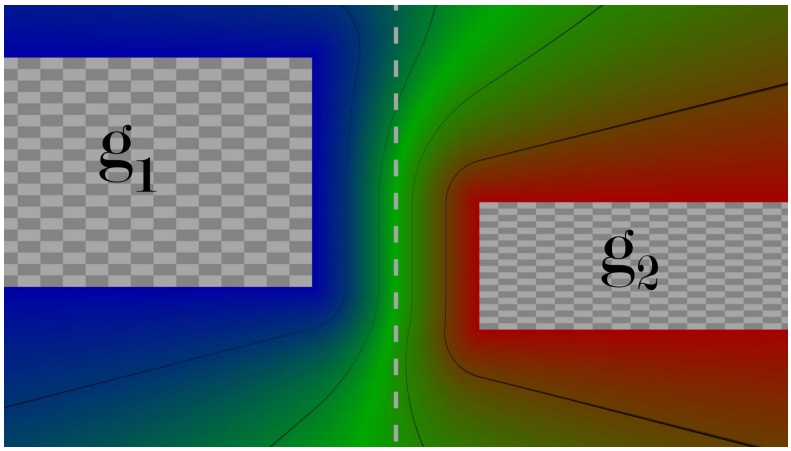


Figure 3.19: *The half-cylinders g_1 and g_2 are the known material* features *in space time. The time-interval is filled with intermediate materials according to an interpolation scheme called the transfinite interpolation.*

The basic formulation for the attribute interpolation is as follows:

$$\mathbf{c}(t) = w_1(t) \cdot \mathbf{c}_1 + w_2(t) \cdot \mathbf{c}_2 \tag{3.22}$$

where $0 \leq t \leq 1$, and the constraints on the weighting functions provide partition of unity:

- $w_1(0) = 1$ and $w_2(0) = 0$ for $f_1 \geq 0$ (on the surface and the interior of G_1)
- $w_1(1) = 0$ and $w_2(1) = 1$ for for $f_2 \geq 0$ (on the surface and the interior of G_2)
- $w_1 + w_2 = 1$.

The simplest linear interpolation is arrived at by assigning $w_1(t) = 1 - t$ and $w_2(t) = t$, but this is a simplistic solution that does not take any particular features of the geometry into consideration and results in counter intuitive attributes in the transitional stages.

This problem is similar to material interpolation in space for *feature*-based heterogeneous volumetric modelling of functionally graded materials. The transfinite interpolation as presented in Biswas *et al.* (2004) was applied to fill the *gaps* in an object with intermediate material properties

or attributes.

The transfinite interpolation is extended to the 4D case of space time and applied in order to define time-variant volumetric material properties. Two space time material features are set where the *gap* is in the time interval $0 \leq t \leq 1$:

$$g_1(\mathbf{p}, t) = f_1(\mathbf{p}) \wedge_0 (-t) \tag{3.23}$$

$$g_2(\mathbf{p}, t) = f_2(\mathbf{p}) \wedge_0 (t - 1) \tag{3.24}$$

where $\wedge_0$ is an R-function for the set intersection. Here the function g_1 defines a space time half-cylinder (semi-infinite cylinder) with the boundary $t = 0$. In the other words, the initial object G_1 exists at any time $t \leq 0$ and then disappears. Similarly, the function g_2 defines a space time half-cylinder with the boundary $t = 1$, which means the target object G_2, appears at the time $t = 1$ and exists for any $t \geq 1$ (see Figure 3.19). By analogy with the 3D weighting expressions of Biswas *et al.* (2004), the transfinite interpolation is applied to these 4D space time material features and thus define the weighting functions $w_1(X) = w_1(\mathbf{p}, t)$ and $w_2(X) = w_2(\mathbf{p}, t)$ using a normalization of each of the inverse defining functions:

$$w_1(X) = \frac{\frac{1}{g_1(X)}}{\frac{1}{g_1(X)} + \frac{1}{g_2(X)}} = \frac{g_2(X)}{g_1(X) + g_2(X)} \tag{3.25}$$

$$w_2(X) = \frac{\frac{1}{g_2(X)}}{\frac{1}{g_1(X)} + \frac{1}{g_2(X)}} = \frac{g_1(X)}{g_1(X) + g_2(X)} \tag{3.26}$$

Note that $g_1(X) = 0$ for $t = 0$ on the surface and at internal points of G_1 and $g_2(X) = 0$ for $t = 1$ on the surface and at internal points of G_2, thus the weighting functions w_1 and w_2 satisfy the above constraints of partition of unity for the interval $0 \leq t \leq 1$.

Figure 3.20 illustrates the application of the above technique to the

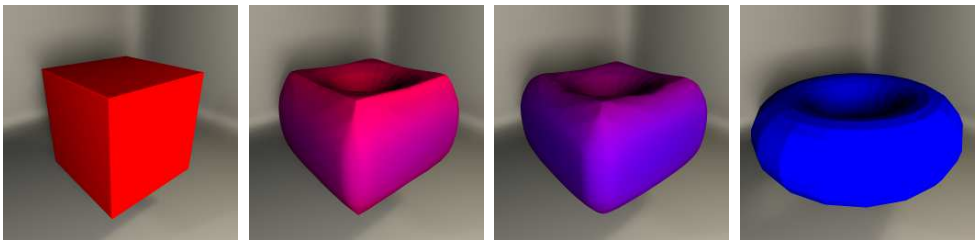


Figure 3.20: *Space time transfinite interpolation of colours during the metamorphosis.*

colour interpolation in the process of metamorphosis between two simple shapes. The colours are constant for the initial objects and represent their homogeneous material properties. From this figure it is apparent that the intermediate colour behaviour of the metamorphosing object is non-linear and geometry dependent in its nature. The colour of the intermediate surface converges faster to the target colour (depicted in blue) when this surface is closer to the target geometry. Similarly, source features far from any of the target surface points remain closer to the original source properties (depicted in red).

Note also that the presented formulation for the space time transfinite interpolation can be easily extended to any material properties function of space. $\mathbf{c}_1$ and $\mathbf{c}_2$ can be rewritten as $\mathbf{c}_1(\mathbf{p})$ and $\mathbf{c}_2(\mathbf{p})$. More precisely, this means that solid texturing can be used, as well as any function providing values of material property attributes at any point in space.

3.4.6 Partitioned objects

In this subsection, we extend the formulation to partitioned objects. In the previous subsection, we assumed that c_1 and c_2 were constant attributes.

Let one of the the objects G_i have multiple semi-disjoint partitions with a constant attribute $\tilde{\mathbf{c}}_j$ assigned to j^{th} partition with the defining function $\tilde{f}_j(\mathbf{x})$. Examples of partitioned objects are shown in Figure 3.21. The overall contribution of the object to the attribute value at a

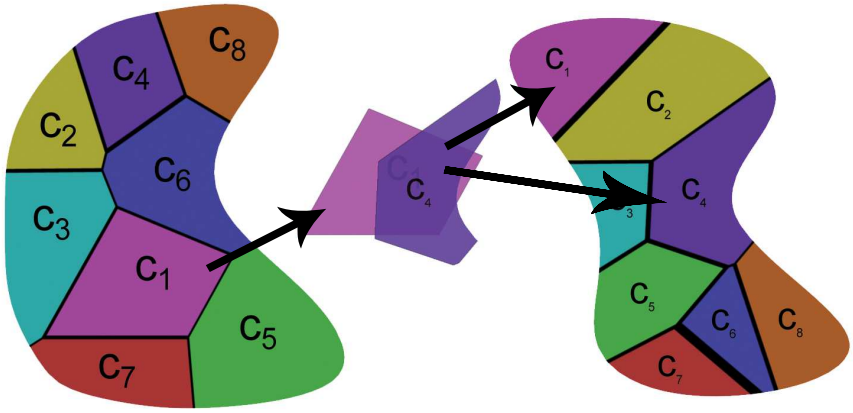


Figure 3.21: *Two partitioned objects with multiple corresponding material features (partitions) indicated by corresponding colours and the case of the overlapping partitions at an intermediate stage.*

given external point $\mathbf{p}$ can be represented by:

$$\mathbf{c}_i(\mathbf{p}) = \frac{\sum_{j=1}^{N} \tilde{w}_j(\mathbf{p})\tilde{\mathbf{c}}_j}{\sum_{j=1}^{N} \tilde{w}_j(\mathbf{p})} \tag{3.27}$$

where N is a number of partitions, and the weight of each partition $\tilde{w}_j(\mathbf{p}) = \frac{1}{\tilde{f}_j^2(\mathbf{p})}$.

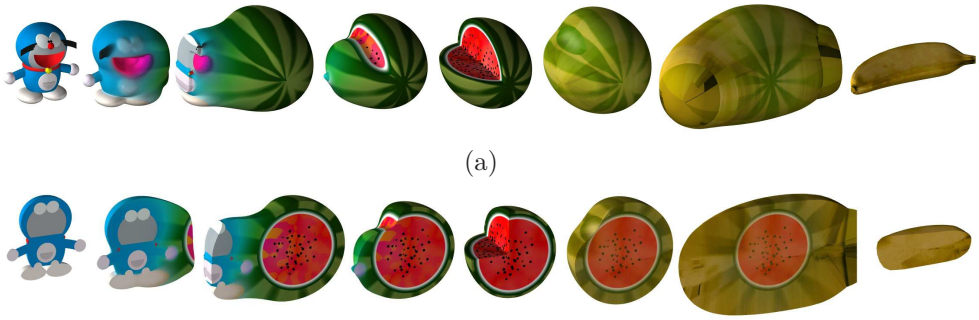
For example, if G_i is represented by a voxel array with an attribute such as color assigned to each voxel, then this expression defines a summation by all the voxels and $\tilde{f}_j$ represents the distance from the given point to the voxel center.

If the number of partitions becomes infinite, i.e., an individual attribute value $\tilde{\mathbf{c}}_j(\tilde{\mathbf{p}})$ is assigned at each point $\tilde{\mathbf{p}}$ of the object point set Ω_j, the above finite summation is transformed into integration as follows:

$$\mathbf{c}_i(\mathbf{p}) = \frac{\int_{\Omega_i} \tilde{w}_i(\tilde{\mathbf{p}})\tilde{\mathbf{c}}_i(\tilde{\mathbf{p}})d\Omega}{W(\mathbf{p})} \tag{3.28}$$

where

- $W(\mathbf{p}) = \int_{\Omega_i} \tilde{w}_i(\tilde{\mathbf{p}})d\Omega$
- $\tilde{w}_i(\tilde{\mathbf{p}}) = \frac{1}{d^2(\mathbf{p},\tilde{\mathbf{p}})}$



(a)

(b)

Figure 3.22: *Metamorphosis with time-variant volumetric materials starting from a cartoon character to a cut watermelon, and then to a banana using the proposed space time transfinite interpolation.*

$d(\mathbf{p}, \tilde{\mathbf{p}})$ being the distance between the points $\mathbf{p}$ and $\tilde{\mathbf{p}}$. The attribute field $\mathbf{c}_i(X)$ can be considered a volume potential with the given density $\tilde{\mathbf{c}}_i(\tilde{\mathbf{p}})$. In fact, the voxel array example mentioned above represents a numerical implementation of this integral formulation.

If both given objects have multiple partitions and there is no established correspondence between the partitions of the two objects, after defining $\mathbf{c}_1(\mathbf{x})$ and $c_2(\mathbf{x})$ using equation 3.27, one can apply equations 3.22-3.26 to interpolate this attribute over time. The volumetric metamorphosis from a cartoon character to a cut watermelon in figure 3.22 is an example of objects with multiple partitions and no established correspondence between them.

In the case when correspondences are established between partitions of two objects as shown in figure 3.21, such that the shape transformation starts with a partition of the initial object and finishes with the corresponding partition of the target object, the equations 3.22-3.26 have to be separately applied to each pair of partitions. In this case, when several transformation processes happen for all the pairs of partitions, the question remains open as to how to combine them into a single intermediate shape with attributes. In general, at intermediate steps partitions may overlap (as shown in figure 3.21) or create disconnected components. Volumetric attributes for the overlapping partitions can be selected following the user defined priorities or some averaging procedure.

3.4.7 User control

In this subsection, additional parameters are introduced to control the effect by affecting the time-interval and the influence of each object. It is important to provide intuitive parameters to control the effects so that an artist or a designer can adjust the effect quickly and easily.

3.4.7.1 Shape driven against time driven interpolation

The previous formulation is not scale invariant. This is because we are using space coordinates (x, y, z) and mix it with time (t). The units do not match. Large objects will lead to a more shape driven interpolation: this is because the distance to the object in 3D space will have a higher impact on g_1 and g_2 than time. On the contrary, small objects will lead to a more time driven interpolation. A shape driven interpolation puts more emphasis on the attributes at the point p, while a time driven interpolation puts more emphasis on the time moment. This is undesirable if it cannot be controlled by the artist. Both behaviours are valid, but the artist needs to be able to control this behaviour. In this subsection we introduce the parameter α to increase or decrease the time gap, which allows the user to control the behaviour of the interpolation.

In order to control the influence of geometry or time for the attribute distribution, the time gap is stretched or shrunk. g_1 and g_2 can be rewritten as follows:

$$g_1(x, y, z, t) = f_1(x, y, z) \wedge_0 (-\alpha t)$$

$$g_2(x, y, z, t) = f_2(x, y, z) \wedge_0 (\alpha t - \alpha)$$

When α is small, the geometry influences the material distribution more than time. On the other hand, when α is large, then we achieve results which are close to linear interpolation. In figure 3.20 the value of α is set to one.

Figure 3.23 shows the material distribution in space at several instances of time during the interpolation process between a blue cube and a yellow torus using different α values. The top row uses $\alpha = 0.1$ which

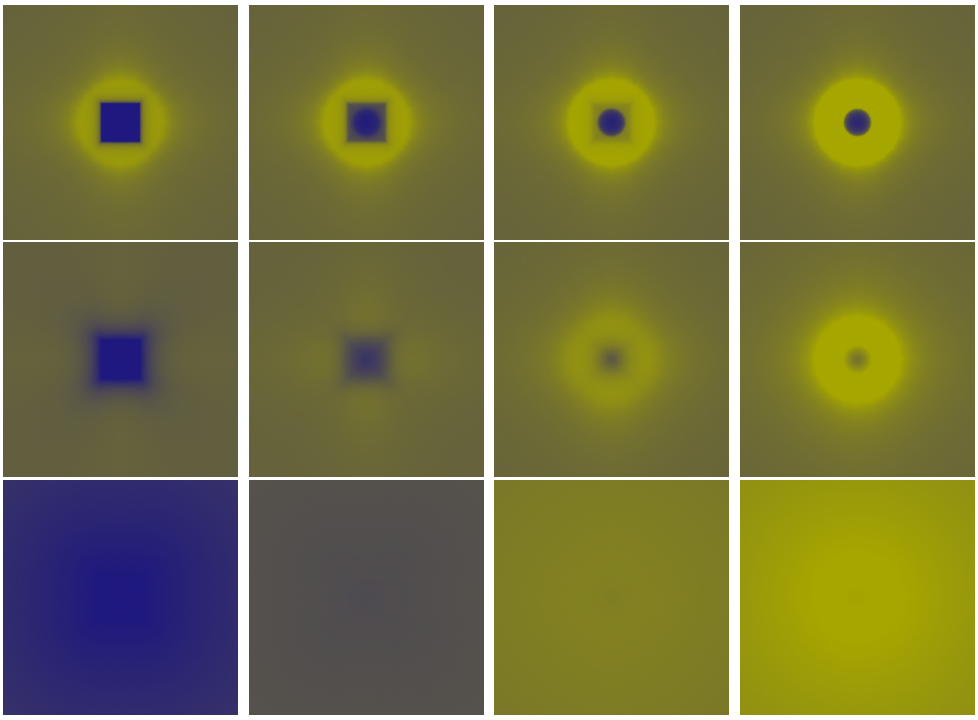


Figure 3.23: *Material distribution using a time gap of 0.1 (top row), 1 (middle row), 20 (bottom row), at time $t = 0$ (first column), $t = \frac{1}{3}$ (second column), $t = \frac{2}{3}$ (third column) and $t = 1$ (fourth column)*

translates to clear geometric features. The interpolation is strongly shape driven. We can see that at $t = 0$, the cube is still blue, but points closer to the torus, elsewhere, are more influenced by the torus colours. The middle row uses $\alpha = 1$ where features are still visible, but the blending happens faster. Finally, the last row uses $\alpha = 20$, making the material distribution almost equivalent to a linear interpolation.

Figure 3.24 shows the metamorphosis between the Stanford bunny and the Utah teapot with two different values for α. The top row shows a small time gap, and therefore, the influence of the geometry is clearly visible. The bottom row shows a large time gap, which leads to a mostly time driven interpolation, which is almost equivalent to a linear interpolation.

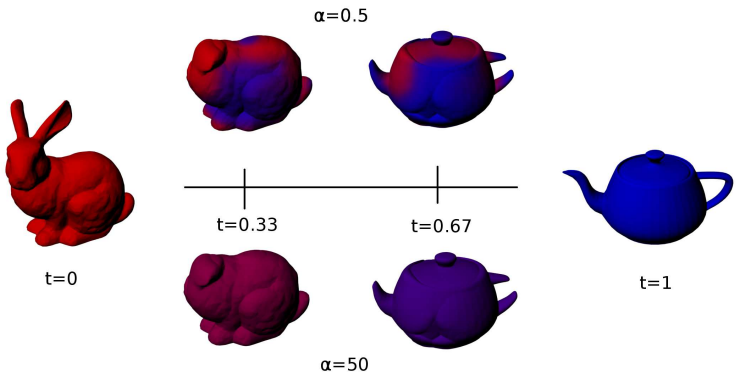


Figure 3.24: *A metamorphosis with different time gap values. The top transformation uses* $\alpha = 0.5$ *and the bottom row* $\alpha = 50$

3.4.7.2 Object influence

In this subsection, we introduce an additional parameter to increase the influence of one object material distribution or the other. It is a bounded parameter, making it easy for the user to try and see the result.

The parameter β is given a real value in the range $[0, 1]$ which is used to scale the values of the object functions. The function g_1 and g_2 are rewritten once more as:

$$g_1(x, y, z, t) = (f_1(x, y, z) \cdot 2(1 - \beta)) \wedge_0 (-\alpha t)$$

$$g_2(x, y, z, t) = (f_2(x, y, z) \cdot 2\beta) \wedge_0 (\alpha t - \alpha)$$

Figure 3.25 shows how the balance parameter can affect the colours. The armadillo in brown is morphed with a gargoyle in grey (top row). The bottom row shows the same intermediate shape, using different balance parameters. This process can be viewed as a scheduling task. The first frame and last frame of the animation will match the source and target object materials. However, the β parameter modifies the weight curves to favour one object providing more artistic control to the animator in an intuitive way.

Figure 3.25: *The influence of the balance parameter on the metamorphosis between two objects. The parameters for the bottom row, from left to right are* $\beta = 0.2$, $\beta = 0.4$, $\beta = 0.6$, $\beta = 0.8$. *The metamorphosis parameter* t *is set to* 0.5

3.4.8 Summary

In this section, the problem of interpolation of volumetric material properties over time was investigated. Unlike most alternatives, this novel method provides a closed form solution which is also shape driven. It relies on an extension of transfinite interpolation to space time to provide a weighting of the attributes in function of space. This method also provides intuitive parameters giving more control to the user. The applications and results are shown in section 4.4. The work presented in this section was published in Sanchez *et al.* (2014).

3.5 Morphological shape generation

In this section, morphological shape design is investigated as yet another use of distance based modelling. The objective is to create new shapes given a set of shapes which will take various parts of various objects and mix them together. Here, several scalar fields are combined with various weights. The weights are based on the distance to various parts of the object.

3.5.1 Problem description

The problem of morphological shape design can be defined as a search for new shapes from a particular application domain represented by a set of selected shape instances. In this case, we would like to be able to take any number of shapes, and *breed* them by selecting and mixing parts of the object. The process is user-driven (i.e., the user defines which parts of which objects should be mixed). As Bar-Zeev (2012) predicts, real changes in design processes will come *"when we can take two models and say, make A more like B, right here in this part but not that other part. If we solve that, then we can imagine a real open ecosystem for 3D designs that truly credits (and rewards) the creators of original designs while allowing easy mashups of the results."* As the first step, being able

to 'breed' a new shape from two given objects, where the user can select which traits of the parents are passed onto the child is acceptable, if the breeding can be performed over several generations. This way, the user can create a genealogy tree of a new shape. However, it would be preferable if the designer would be able to select traits from an arbitrary number of parents, and control the final shape through a single, simple interface. This would permit the designer to directly input the desired weights of each trait directly.

3.5.2 Criteria of evaluation

The method we seek should have several properties to enable the artists or designers to achieve their visions. First, the method must be able to produce objects in reasonable time to allow artists and designers to quickly generate and adjust parameters. A lot of objects need to be generated during the search, and so the process should not take much time to show the results. Then, there should be no restrictions on the number of initial shapes. As stated before, a binary tree is an acceptable solution, provided that simple controls can be provided. The final shape should be easily controlled by a non-technical person, therefore, it should be done through simple and intuitive parameters. Next, the weighting of each initial shape influence has to be provided when mixing the various parts of the objects. Finally, the control must be provided to ensure a single-component object as the result, not a set of disconnected components (e.g., legs of a chair, seat, arm rests) so that it can be printed as a single object.

We want to underline that in the application of this technique, the designer makes the final decisions on which models to select and whether the process is successful.

3.5.3 Existing approaches

In computer graphics and shape modelling, morphological shape design was approached through genetic algorithms applied to polygonal models (Sims 1994), pre-segmented meshes (Xu *et al.* 2012), algebraic surfaces (Bedwell and Ebert 1999), parametric surfaces (Lamas and Duro) and procedural implicit models (Silva *et al.* 2005). Some of these methods require the user to perform a tedious match of surfaces between shapes, while others require the user to identify the best resulting shapes, as the evolution process itself is completely automatic. The user also has a limited control over the process through the parameters of the cross-over and mutation. An alternative procedural approach was introduced in Kalogerakis *et al.* (2012) for a collection of shapes from an identified complex domain. A probabilistic shape synthesis procedure is applied to a given set of models segmented into compatible components. In Zheng *et al.* (2013), new objects are built from a set of objects in a database, by creating graphs and connecting compatible parts. This method is limited by the sub-structure graph, preventing some objects to be used with others. Finally, those methods create models that usually need to be repaired first before 3D printing, which leads to a manual and time consuming process.

While metamorphosis has not been directly proposed for the purposes of morphological shape modelling, it can be applied several times as a family tree. Section 2.4.2 provides more details on the various methods for metamorphosis. However, most metamorphosis algorithm based on boundary representations often impose a lot of limitations on the shapes that can be morphed together. There are some exceptions, but they require user interactions (DeCarlo and Gallier 1996; Takahashi *et al.* 2001). Scalar field or voxel-based methods do not have such limitations. Out of those, most can only be performed on two models at a time (Pasko *et al.* 1995; Cohen-Or *et al.* 1998; Turk and O'Brien 1999a; Pasko *et al.* 2004), and do not allow for various parts to be interpolated at different speeds. More user control can be achieved by restricting the class of objects. Thus, in Wyvill (1993) some control over the metamorphosis of skeletal

soft object was achieved by cellular matching or hierarchical matching and the metamorphosis was performed by interpolating the positions of the skeletons and the field intensities. The main drawback of this method is the requirement of skeleton matching for two objects which is similar to the shape matching problem defined for polygonal models. Later in Galin *et al.* (2000) this limitation was partially lifted, however the matching between the structure of two objects was still the issue. Metamorphosis control was improved by Lerios *et al.* (1995), where the user defines pairs of corresponding features in the source object and the target object and to interpolate between the shapes of the source model and the target model by using the information about these features. Features were defined by the user, as an oriented bounding box, a rectangle, a line or a point.

Group metamorphosis was addressed for limited groups of objects with simple weighting schemes such as barycentric coordinates for three objects Adzhiev *et al.* (2005) and the bilinear interpolation for four objects Fausett *et al.* (2000). Both of these methods do not perform component-based interpolation of objects, preventing the user from choosing the traits to interpolate, and sometimes leading to disappearing components.

3.5.4 Overview

In contrast to existing generative procedures, an approach based on a user-controlled metamorphosis between functionally based shape models is presented. A formulation of the pairwise metamorphosis is proposed with a variety of functions described for the stages of deformation, morphing and offsetting in section 3.5.5. This formulation is then extended to the metamorphosis between groups of shapes with user-defined, dynamically correlated and weighted feature elements in section 3.5.6.

3.5.5 User-controlled pairwise metamorphosis

Let A and B be the source and target objects respectively, and their real-valued function representation be the functions $f_a(\mathbf{p})$ and $f_b(\mathbf{p})$. While there are no restrictions on these functions apart from C^0 continuity, it is preferable that these functions have distance properties. A function with distance properties decreases with the distance to the zero-level set and increases with the distance. Amongst other reasons, offsetting and metamorphosis will be used and generally behave better with distance fields. C^1 continuity can also improve the smoothness of the transition if smoothness is required. In this section, convolution filtering of distance fields (see section 3.2) or signed distance fields is used for f_a and f_b.

Lerios *et al.* (1995) defined the notion of *feature elements*, which are used to establish the correspondence between parts of the objects. A *feature element* can be described as follows:

- The *feature element* i is represented by a signed distance function $d_i(\mathbf{p})$;
- The *feature element* encloses the spatial area of the object which is part of a feature of the object which has a correspondence, and that will be deformed;
- Each *feature element* E_a of the object A corresponds to the feature element E_b of the object B and there exists a bijective mapping T that maps any point $\mathbf{p}_A \in E_a$ to the point $\mathbf{p}_B \in E_b$ and an inverse mapping T^I that maps the point $\mathbf{p}_B$ back to $\mathbf{p}_A$.

The user is required to define n feature elements on the source object and their corresponding n elements on the target object. The feature elements do not need to be placed accurately and only need to roughly describe the spatial area of the feature. Figure 3.26 illustrates the feature matching process where the user defined eight features on both chairs. Each feature i has an associated weight function $w_i(d_i(\mathbf{p}, t))$ which evaluates the importance of the feature's function at time t for the given point. The weighting function will be discussed in section 3.5.5.3.

Figure 3.26: *Matching features with different types of geometry for feature elements: for legs cylinders are used, for spine a prism-like polytope.*

The metamorphosis function f_m represents some type of smooth interpolation between the function f_a defining the initial shape at t=0 and the function f_b defining the target shape at $t = 1$. The metamorphosis function $f_m(f_a, f_b, t)$ is chosen such that $f_m(f_a, f_b, 0) = f_a$ and $f_m(f_a, f_b, 1) = f_b$ where t is a parameter to define the evolution of the morphing object, $t \in [0, 1]$. Several scalar field based metamorphosis functions can be used (see section 2.4.2). More details are provided in section 3.5.5.1. Additionally, an offset function $o_i(t)$ defined for each feature is introduced in section 3.5.5.4 to control the behaviour of the features during the metamorphosis. This offset is used to balance a common issue with scalar field based metamorphosis functions where the shape disappears during the intermediate frames or undesired disconnected components appear. The general, pairwise, user-controlled

metamorphosis function can be formalized as follows:

$$F_M(\mathbf{p},t) = \frac{\sum_{i=1}^{n} w_i(d_i(\mathbf{p},t))\left(f_m(f_a(\mathbf{p}_a^i), f_b(\mathbf{p}_b^i), t) + o_i(\mathbf{p})\right)}{\sum_{i=1}^{n} w_i(d_i(\mathbf{p}))} \tag{3.29}$$

where:

- f_m is a metamorphosis function. It can be a trivial linear interpolation, but a more complete description is given in section 3.5.5.1
- d_i is a function providing a distance to the feature i at the given time t. The details of this function are given in section 3.5.5.2
- w_i is a weighting function which evaluates the importance of each feature for the point $\mathbf{p}$. Details are provided in section 3.5.5.3.
- o_i is an offset function which locally grows parts of the object to overcome disconnected components. Details are given in section 3.5.5.4

Here $\mathbf{p}_a^i$ denotes the point in the object A and $\mathbf{p}_b^i$ denotes the point in the object B corresponding to the point $\mathbf{p} = (x, y, z)$ in the frame i by using forward and inverse transformation mentioned above.

The function F_m represents the final object, where the surface is defined by $F_m = 0$, and the interior of the object is defined by $F_m > 0$. The object can be visualized using ray-marching. A manifold mesh can be extracted with the marching cubes technique (Lorensen and Cline 1987) if necessary. There are other meshing techniques available, such as Ju *et al.* (2002), Kobbelt *et al.* (2001) and Nielson (2004). However, we prefer Lorensen and Cline (1987) since it guarantees the mesh extracted will be manifold.

In short, the process can be seen as the application of the following scalar field operations: weighted average of the morphed objects after transformations (space mappings) and offsets (see 3.27).

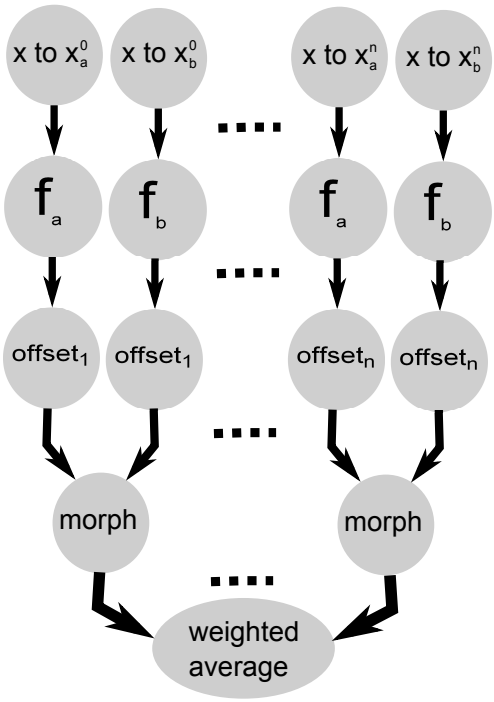


Figure 3.27: *The process of the pairwise metamorphosis with user-defined and weighted features.*

3.5.5.1 Morphing functions

Section 2.4.2 provides an overview of metamorphosis functions. The linear metamorphosis often leads to disconnected components, and it is preferable to avoid it. The function presented in Pasko *et al.* (2004) provides better results, with more interesting shapes and less disconnected components. However, for any metamorphosis function, the user can adjust the parameter t to accelerate certain phases of the metamorphosis. This can be useful if a function has a more interesting time interval where the shape changes too quickly.

3.5.5.2 Space mappings

The forward mapping T transforms the given point $\mathbf{p}$ into $\mathbf{p}_b^i$ with respect to feature i for the target object B and inverse mapping T^I transforms the point $\mathbf{p}$ at frame i into $\mathbf{p}_a^i$ with respect to feature i for the source object A. Some of the mappings that can be used are: affine mapping, twist, taper, bending. Those mappings can be combined by applying

(a) (b)

Figure 3.28: *Deformation without metamorphosis: (a) the original chairs A and B, (b) chair A under deformation of B (left) and chair B under deformation of A (right)*

them consecutively.

For this metamorphosis application, the mapping needs to have a continuous interpolation between the source and target values. Most parameters can use linear interpolations (e.g. translations, scaling, twist angle) while others require more work (e.g. spherical linear interpolation for rotations). Following these definitions, the mappings T and T^I are parametrized by a time dependent parameter $p(t)$. Thus, $T = T(\mathbf{p}, p)$ and $T^I = T^I(\mathbf{p}, p)$ where $p(0) = p_a$ and $p(1) = p_b$ are values of the parameter in the source and destination objects. In conclusion, the points $\mathbf{p}_a$ and $\mathbf{p}_b$ can be obtained as follows:

$$\begin{aligned} \mathbf{p}_a &= T^I(\mathbf{p}, p(t)) \\ \mathbf{p}_b &= T(\mathbf{p}, p(1-t)) \end{aligned} \tag{3.30}$$

The simplest case of the mapping function is the affine transformation. In this case, the forward mapping is defined by a transformation matrix, the inverse mapping is defined by the inverse of this transformation matrix. Transformation matrices can be linearly interpolated as shown in Shoemake and Duff (1992). Alternatively, instead of matrices, quaternions and spherical linear interpolation can be used for the rotations, and linear interpolation is applied on translation and scaling.

Finally, the time parameter t for the deformation process (space mapping) and metamorphosis can be independent. For instance it is possible to deform an object but not use the metamorphosis. In figure 3.28, the

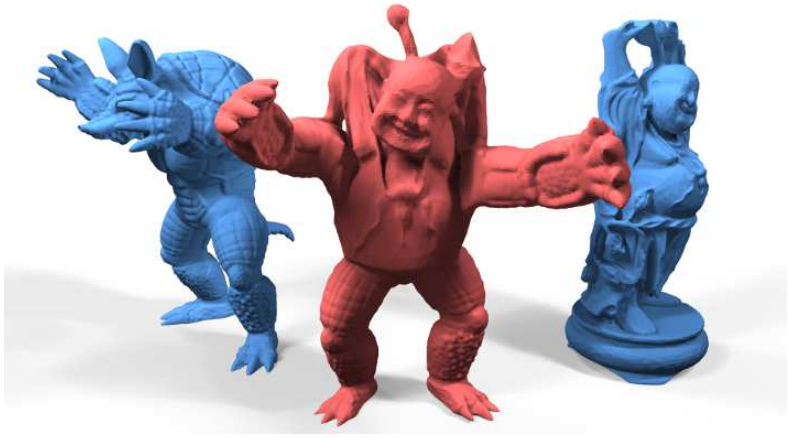


Figure 3.29: *Morphing and deformation parameters can be independent of each other, and dependent on space. Here parameters are feature-based.*

original chairs are on the left. On the right, both chairs have been deformed to match each other's feature orientation. In this example, only the deformation was applied, but the metamorphosis was not applied. That is, each feature of the model is kept as it is albeit moved, rotated and scaled to match the other object's feature transformation. Additionally, the parameter t can be set per feature element. This results in more control allowing the user to choose which parts of the object should deform and morph into the other.

The figure 3.29 shows the use of this technique between the armadillo model and the Buddha model from the Stanford repository. In this example, six features per object are defined by the user (left and right legs, left and right arms, main body and head). The user defined those features using oriented boxes. The morphing parameter is set per feature to take elements of the Armadillo (blue shape on the left) model and elements of the Buddha (blue shape on the right) model and combining them together. Some parts are blends of the feature pairs. The red shape (the monster) uses the arms and legs of the armadillo model, the head of the Buddha, and a blend of both bodies.

3.5.5.3 Weighting functions

The previous sections covered deformation and metamorphosis, but each of those operations must be performed per feature, and correctly weight each value with respect to $d_i(\mathbf{p})$. In Lerios *et al.* (1995), the square inverse of the distance to the feature is used as the feature weight:

$$w(d) = \frac{1}{(d+\varepsilon)^2} \tag{3.31}$$

This function however is not easy to control because of the ε value, and is not smooth at $d = 0$. It causes visible edges close to the feature boundaries.

Instead, we modify slightly the formula for the transfinite interpolation of n source (Biswas *et al.* 2004) to support features that can overlap at the transition stage:

$$w_i = \begin{cases} \dfrac{\max(0,d_i)}{\sum_{j=1}^{n} \max(0,d_j)} & \max_i(d_i) > 0 \\ \dfrac{\prod_{j=1;j\neq i}^{n} d_j}{\sum_{j=1}^{n} \prod_{k=1;k\neq j}^{n} d_k} & otherwise \end{cases} \tag{3.32}$$

The transformation in Lerios *et al.* (1995) was a simple affine transformation. For each feature pair (E_a, E_b), both objects were moved so that the two features would align with each other in the intermediate frames, and then the metamorphosis was applied. If there are n feature pairs, then n metamorphosis are applied. This process is repeated for all the features, and then a weighted average of the field is applied, where the weight is a function of space. This produces a simple deformation (as seen in figure 3.28). Figure 3.28a shows the two original objects, and figure 3.28b shows each object deformed so that their features are aligned with the features of the other object.

3.5.5.4 Offsets

Thin features can be problematic, especially for 3D printing. Issues can arise at many stages:

- The function F_m may have disconnected components, if the metamorphosis is applied on objects with thin features.
- During the mesh extraction, grid based methods such as Lorensen and Cline (1987), Ju *et al.* (2002) or Nielson (2004) may miss features which are smaller than the grid resolution.
- After printing, if the features are too thin, it may break.

While the previous operations greatly help to reduce the chances of disconnected components, additional operations may be needed. A general offset can be applied to the final field to reconnect the disconnected components. However, this tends to blur out the details when the offset is not needed. A better approach is to define a new scalar field to control the offset value. This reconnects the disconnected components only when it is needed. Such a field is not trivial to define, especially for designers. A simpler solution is to define an offset per feature. This offset can simply be defined as follows:

$$o_i(\mathbf{p}) = d_i(\mathbf{p}) + d \tag{3.33}$$

The offset can be applied on top of the metamorphosis operation, which is later weighted by the user-defined features. Therefore, the user can define the offsets if needed, per feature. In the chair example in figure 3.26, a designer could set small offsets for the legs of the chairs. If no offset is needed, they are set to zero, and will have no impact on the equation 3.29.

3.5.6 User-controlled group metamorphosis

In the previous section, shapes were designed using a user-controlled metamorphosis between two objects only. Although, this pairwise meta-

morphosis can be applied iteratively to a group of objects, a generalization of the proposed formulation that creates complex shapes directly from a group of objects in the design of a new shape is more attractive.

In this section, we present the modifications to the formula in equation 3.29 in order to support any number of objects. Given k objects where the j-th object, with $1 \leq j \leq k$, is defined by a signed distance function $f_j(\mathbf{p})$ with the morphing function $f_m(f_1, f_2, ..., f_k)$ and n feature elements defined for each object such that the signed distance function for the feature i of the object j is $d_{j,i}$, the user-controlled group metamorphosis function becomes:

$$F_{M_k}(\mathbf{p}, t) = \frac{\sum_{i=1}^{n} w_i(d_i(\mathbf{p}))\left(f_m(f_1(\mathbf{p}_1^i), ..., f_n(\mathbf{p}_n^i)) + o_i(\mathbf{p})\right)}{\sum_{i=1}^{n} w_i(d_i(\mathbf{p}))} \tag{3.34}$$

Unfortunately, this new formulation requires the morphing function f_m to take any number of arguments. This is not the case of space time blending (Pasko *et al.* 2004), which can only morph between two objects at a time. Therefore, the only suitable function for morphing is the linear interpolation, which has already been used in the past to interpolate between several objects (Adzhiev *et al.* 2005). Using linear interpolation, F_{M_k} is rewritten as follows:

$$F_{M_k}(\mathbf{p}, t) = \frac{\sum_{i=1}^{n} w_i(d_i(\mathbf{p}))\left(\sum_{j=1}^{k} f_j(\mathbf{p}_j^i)v_{j,i} + o_i(\mathbf{p})\right)}{\sum_{i=1}^{n} w_i(d_i(\mathbf{p}))} \tag{3.35}$$

where $v_{j,i}$ defines the weight or the influence of the feature element i of the j-th object in the group metamorphosis process with $\sum_{i=1}^{n} v_{j,i} = 1$.

The available space mapping functions are also more difficult. While most space mapping parameters can still be used through weighted sums, orientation cannot. The spherical linear interpolation on quaternions

and the matrix rotation interpolation do not extend well to the weighted average of k elements. Some methods exists, for instance, presented in Moakher (2002), however this may not behave as the designer would expect. Some examples are shown in section 4.5.

3.5.7 Summary

In this section, a novel method to combine different parts of different models into a new single object was presented. The method relies on a weighting field which is controlled by several distance fields. This weighting field is used to apply independently deformation, morphing and offsets, per shape feature. The formulation was first introduced for two objects to identify the key components of the morphological shape generation, and then extended to any number of objects. The applications and results are shown in section 4.5. The work presented in this section was published in Sanchez *et al.* (2013).

3.6 Conclusions

In this chapter, a new theoretical framework was introduced and several distance based heterogeneous volumetric modelling problems were tackled.

- The convolution filtering presented in section 3.2 served as a foundation by introducing an approximate smooth distance field given any distance field. C^1 continuity of the approximate distance field is important for many applications and not only restricted to heterogeneous volumetric modelling.
- A novel way for the volumetric interpolation of the material properties of an object was introduced in 3.3. Unlike alternative *feature* based heterogeneous volumetric modelling techniques, the presented method is shape aware, and considers the *feature source* accessibility in the interpolation process.

- The problem of interpolating in time between the material distributions of two objects was tackled in section 3.4. This section dealt with the time-dependent changes to the volumetric material properties as a transformation of the volumetric material distributions in space time accompanying geometric shape transformations. Smooth distance fields are a key element of this technique.
- Finally, distance fields were used to combine several fields to achieve morphological shape generation in section 3.5. This section highlighted the relation between property functions (scalar fields of properties or attributes) used as a parameter for the geometric modelling process.

In the next chapter, the technical details and algorithms for these four contributions will be described.

Chapter 4

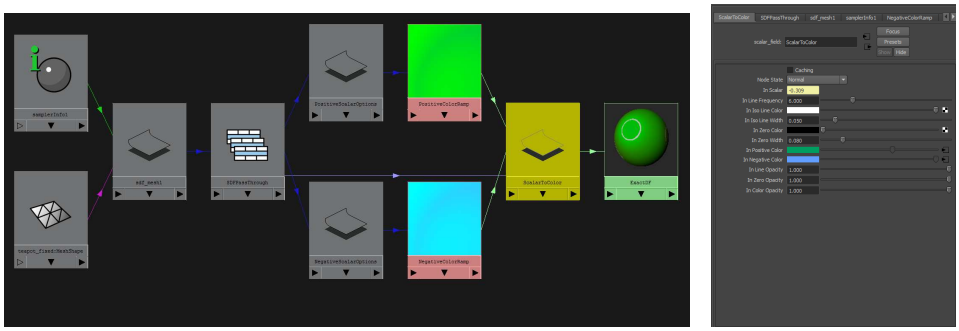
Applications and results

In this chapter, applications of the techniques introduced in chapter 3 are shown. First, in section 4.1, the details of how this work was implemented and how users can interact with the system are shown. In section 4.2, convolution filtering applications for shape modelling and attribute interpolation are shown. In section 4.3, results of the shape aware volumetric interpolation for material and microstructure control are shown and explained. Finally in section 4.4, a few applications of space time transfinite interpolation are detailed.

4.1 Implementation and user interface

Depending on the type of user, there can be several options to use the techniques presented in this chapter. An advanced user can implement these technique in C directly, or in a script such as a HyperFun script (Adzhiev *et al.* 1999). Finally, for non technical users, such as an artist or a designer, a graphical interface can be used.

All the techniques presented in this thesis were implemented as a Maya (Autodesk 2014) plug-in and as stand-alone command line applications. Command line applications are useful for developers, but not for users. For user-friendly interfaces, Maya was chosen. There are several reasons for choosing this system:

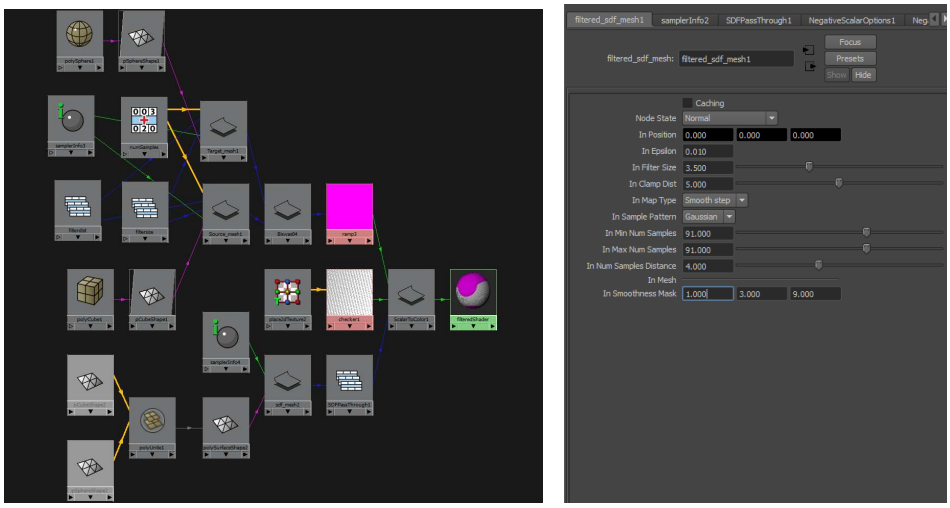


(a) (b)

Figure 4.1: *a) DAG Hierarchy example, a signed distance field is converted to a colour map. b) An attribute view of the ScalarToColour node*

- Maya is a well known modelling package, and many users are comfortable with its interface and concepts.
- The Maya API is flexible and makes the integration with the software seamless.
- Maya features are modular, and therefore, new features introduced in a plug-in can use all the existing features.
- Maya can be used by non-technical users as well as technical users.

Maya relies on an object hierarchy called directed acyclic graph, or DAG. The DAG hierarchy provides a framework to evaluate nodes in a certain order. Each node in the graph has a number of inputs, and a number of outputs. Input and attributes of the nodes are interchangeable in Maya. Any attribute value can be set by another node, or set manually in the attribute interface. Figure 4.1 shows an example of the DAG view in Maya. In this example, a signed distance field is generated from a mesh, and converted to a colour map. The node *sdf_mesh*1 takes in a mesh, and the sampling point, and outputs the signed distance to the mesh from the sampling point. The node *ScalarToColour* outputs a colour based on the distance. Building this graph could be tedious, therefore, Maya lets plug-in have *'commands'* to automate this process. Commands can be used to build these graphs, create objects (such as a mesh), or modify node attributes.



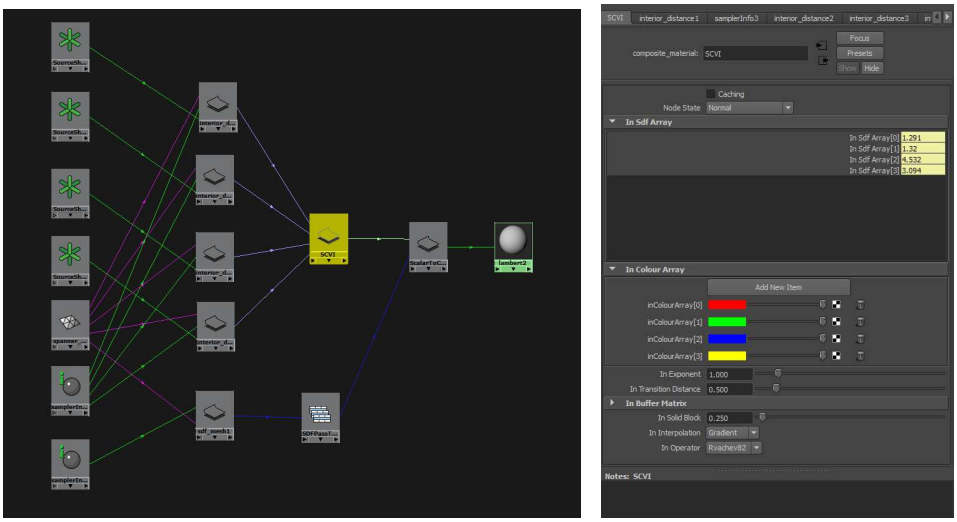
(a) (b)

Figure 4.2: *a) The DAG hierarchy for transfinite interpolation between two features using convolution filtering b) the convolution filtering node attribute view*

Since the techniques presented are mostly functions of point coordinates and sometimes time, all the plug-in nodes require at least the sampling point. These nodes are used for shading. When a mesh or a surface needs to be shown in the modelling window, to be exported or manipulated, a command which creates an isosurface mesh from a function of point coordinates is used. The typical process for a user, is to select a number of elements and call a command, which generates a graph. Once this graph is generated, a mesh can be created through an additional command, or the scene can be rendered by attaching the graph to a shader. The subsequent subsections provide typical user cases.

4.1.1 Convolution filtering use case

There are several applications for convolution filtering. For its intended purpose, smooth interpolation, no geometry needs to be created. The user selects two meshes, and using a command, a convolution filtering node is created for each mesh. Then these two nodes are selected by the user, and another command creates the shader for the interpolation. The figure 4.2a shows the graph generated by the commands, and figure



(a) (b)

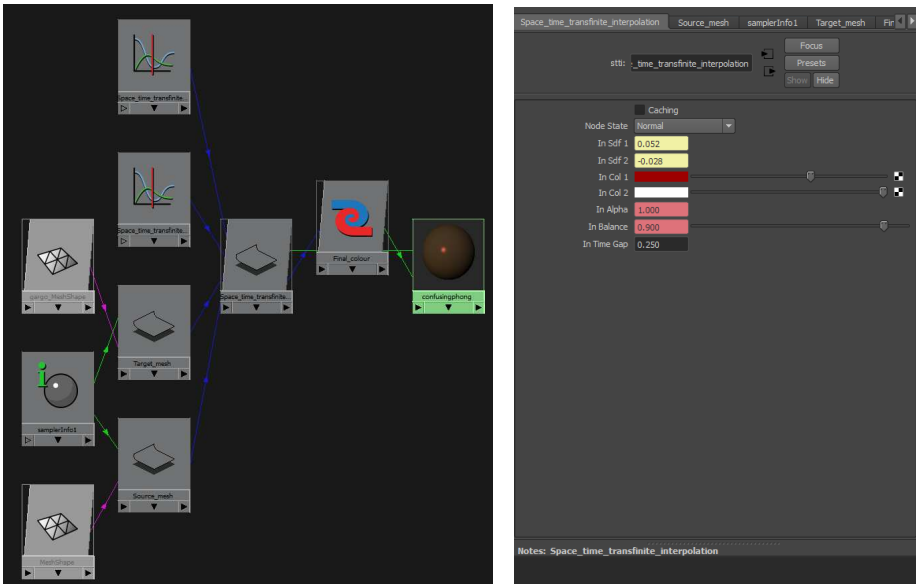
Figure 4.3: *a) The DAG hierarchy for shape conforming interpolation, b) the attribute view of the shape conforming interpolation node*

4.2b shows the attribute view of the convolution filter nodes. The user can then change the capping distance f_c, the maximum kernel size and the adaptive sampling parameters.

For other applications, such as blending unions, the user selects two meshes, and calls a command which generates a new mesh. It is also possible to build the union with a node, and then mesh the object through a command.

4.1.2 Shape conforming volumetric interpolation use case

For shape conforming volumetric interpolation, the user needs to select either one mesh, and any number of locators, or several meshes. If the user selects locators, the material features will be points located where the locators are, and the object will be the selected mesh. If the user selects several meshes, then the first mesh is the object in which the interpolation will occur, and the other meshes will be the material features. The user can then call the appropriate command which builds the graph (shown in figure 4.3a). Each material feature has an associated



(a) (b)

Figure 4.4: *a) The DAG hierarchy for space time transfinite interpolation, b) the attribute view of the space time transfinite interpolation node*

node which produces a distance (interior distance by default). These distances are passed to the shape conforming interpolation node. The shape conforming interpolation node has a number of attributes. For each material feature, a colour is defined. Then the user can change the transition distance in the attribute view, or call a command to calculate the factors $s_{i,j}$ as detailed in section 3.3.7. There are other minor parameters such as the R-Function to use.

4.1.3 Space time transfinite interpolation use case

For space time transfinite interpolation, the user simply has to select two meshes, and call a command. The command can use any type of distance implemented (signed distance fields, convolution filtering of signed distance fields, signed L_p distances). The DAG hierarchy is shown in figure 4.4a. Then the operator parameters (such as balance, time gap and time) can be accessed in the attribute view of the space time transfinite interpolation node (see figure 4.4b).

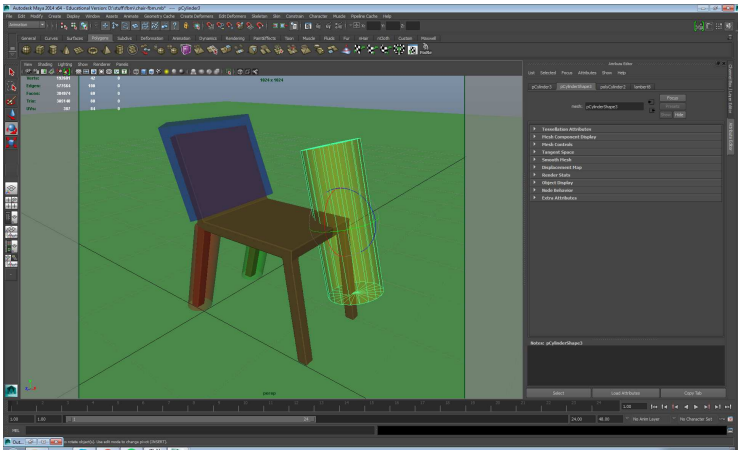


Figure 4.5: *A user defines the feature volumes on an object.*

4.1.4 Morphological shape generation use case

The process for morphological shape generation is a two step process. The first step is to define the features for all the models. Figure 4.5 shows the Maya interface during the process. The back legs and back of the seat are set, the front leg is in the process of being defined. If there are only two models (pairwise metamorphosis, see section 3.5.5), then the user selects the two models and all the features, and then a command will create the intermediate shapes at the desired resolution. If the user wants to use the group metamorphosis (as detailed in section 3.5.6), then all the models, their features must be selected, and the matrix for deformation and metamorphosis needs to be provided through a file or as command arguments.

Outside Maya, the stand alone application is capable of using the objects and their features and generate more models in one run to quickly explore the design space. The resulting object is also exported as a model with features which can be used again in the metamorphosis.

4.2 Applications of convolution filtering

In this section, the potential applications of convolution filtering (see section 3.2 and section 3.2.9) is detailed. The original objective was

to provide a C^1-continuous field of a mesh everywhere in space except the points on the surface. However, other applications such as local smoothing are possible.

First, we compare the results of convolution filtering with signed distance fields and evaluate the smoothness through visual comparison and gradient field edge detection. Then, we compare signed distances and convolution filtering on a number of applications for which convolution filtering is useful. For these applications, we want a smooth interpolation of attributes (section 4.2.2), or a smooth surface (sections 4.2.3, 4.2.4, 4.2.5, 4.2.6). Each application shows that convolution filtering provides a visually smooth surface or attribute interpolation while preserving the exact surface of the original object.

4.2.1 Evaluation

The quality of convolution filtering can be evaluated visually or by applying subsequent operations on top of the resulting fields.

The figures 4.6a and 4.6b show the exact distance field and the approximate distance field of a gear model. The figure 4.6a shows clear discontinuities along the medial axis, while figure 4.6b shows smooth iso-lines as the distance to the surface grows. The figures 4.6c and 4.6d show the gradient fields of the exact signed distance function, and the approximate smooth distance function. The surface of the object is marked with a black outline to facilitate the readability. Again, the signed distance function clearly shows the abrupt gradient variations, while figure 4.6d seems smoother away from the surface.

The visually abrupt changes in the gradient field can be visualized with a Sobel filter. Figures 4.6e and 4.6f show the edges of the figures 4.6c and 4.6d. The approximate smooth signed distance field is smoother even though it still has discontinuities due to the numerical integration. When close to the surface, and on the medial axis, the gradient edge is more visible, which is also expected, as the smoothing kernel gets smaller, and the overall field becomes almost equivalent to a distance

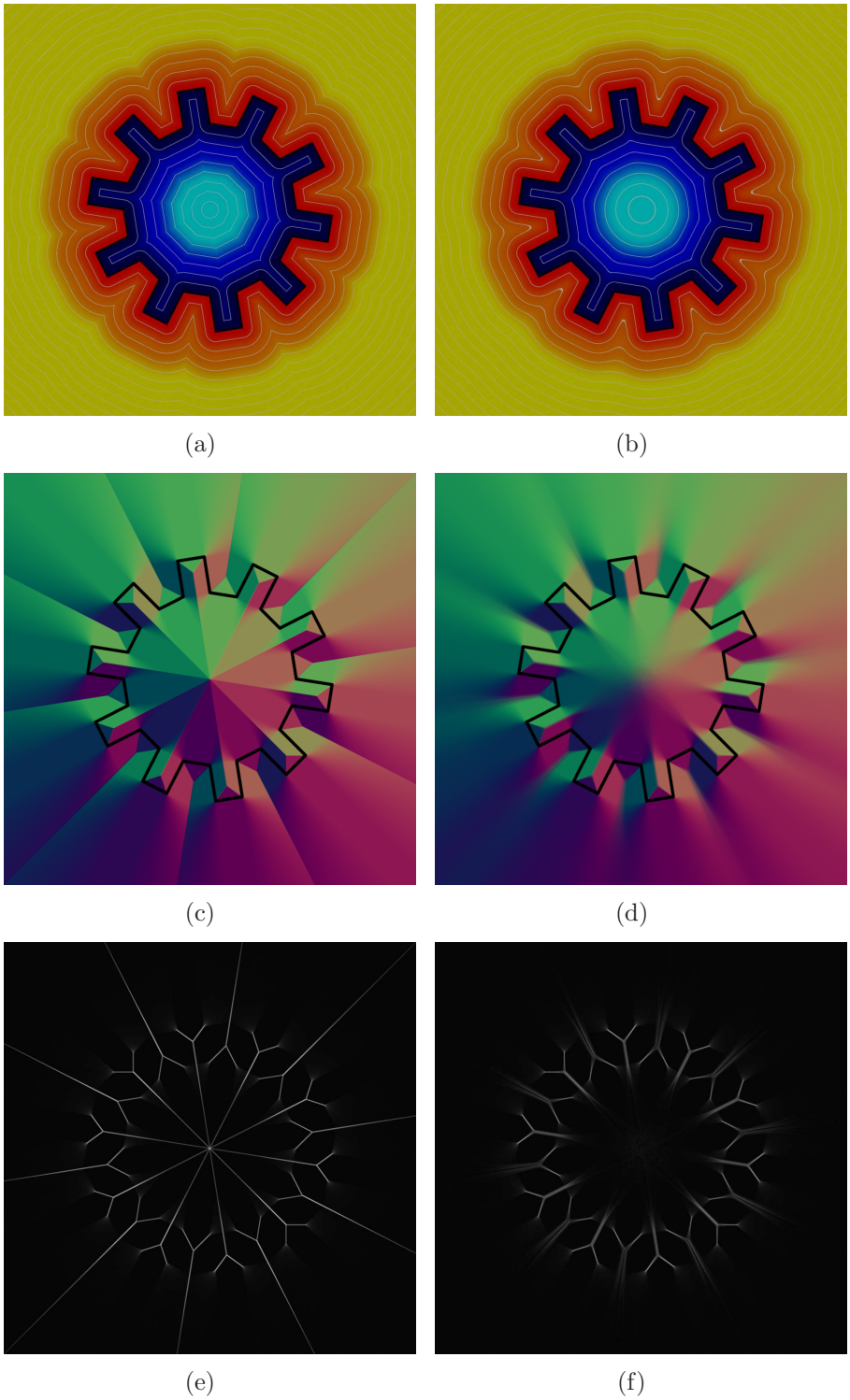


Figure 4.6: *A number of fields produced by a gear model. a) the signed distance field, b) the convolution filtering, c) the gradient field of the signed distance function, d) the gradient field of the convolution filtering function, e) the edges of the gradient field of the distance function revealed by a Sobel filter f) the edges of the gradient field of the convolution filtering function revealed by a Sobel filter*

field. Additionally, the edges in figure 4.6e are whiter than in figure 4.6f, which signifies more visible edges in the exact distance gradient field.

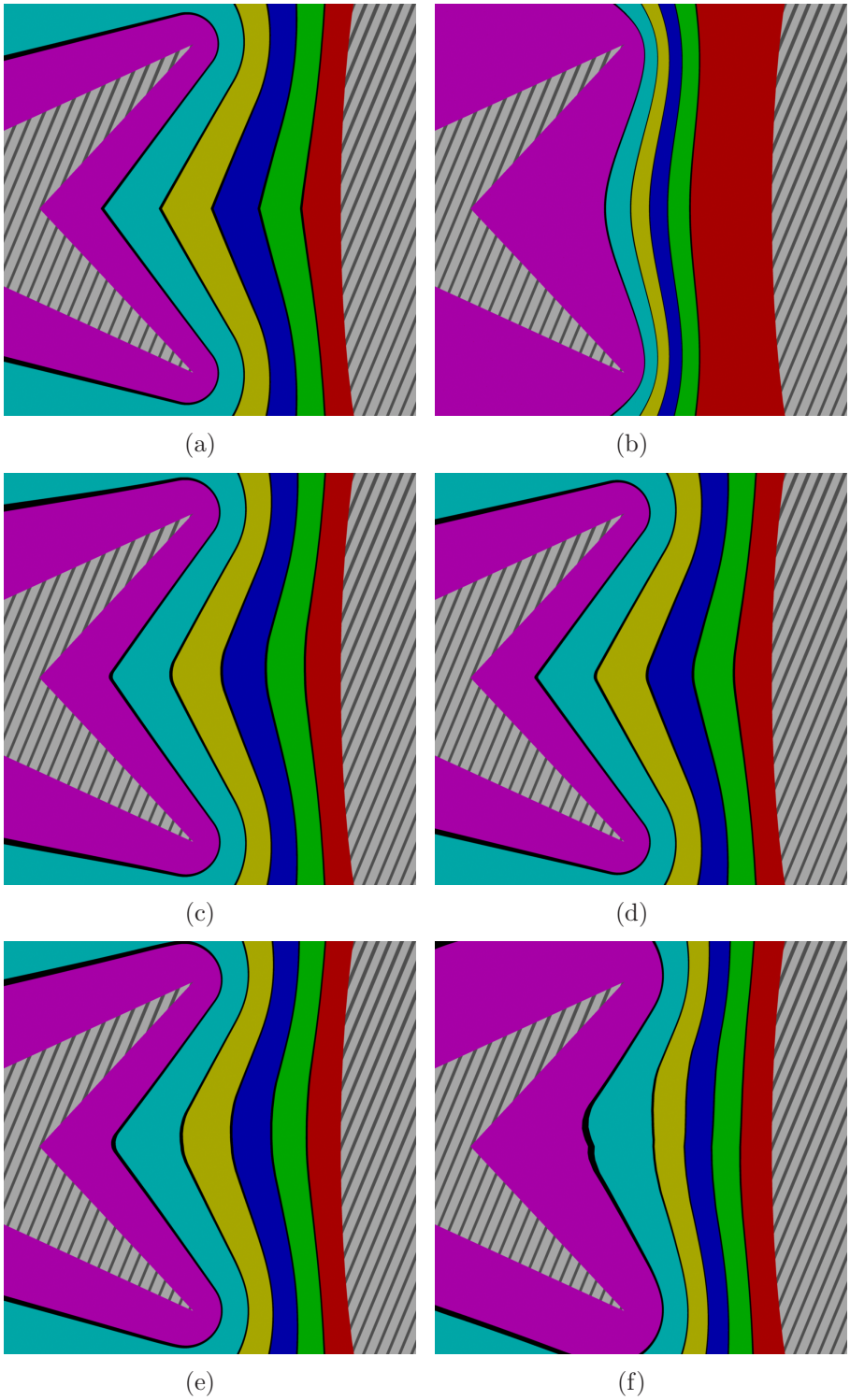
In conclusion, the discontinuities on the medial axis are significantly reduced, and the edges on the gradient field are less visible. Some discontinuities remain only because of the numerical integration, but the approximation improves with more samples.

4.2.2 Transfinite interpolation

C^1-continuous fields are desirable for heterogeneous volumetric modelling. Transfinite interpolation (Rvachev *et al.* 2001) using our filtered field achieve better results than signed distances, and with better control and speed than L_p distances (Belyaev *et al.* 2012). The technique introduced in Rvachev *et al.* (2001) defines material features which have prescribed attribute values. The attributes are then interpolated across space based on the distances to the boundaries of the material features. The formulation relies on distance properties, but the C^1 discontinuities cause stress concentrations and other issues due to the loss of differential properties (Biswas *et al.* 2004; Fayolle *et al.* 2006).

A comparison of convolution filtering with the alternative solutions is given in this section. As stated in the previous chapter, we require the field to exactly represent the material features (including sharp features). The field should also be close to the distance field, and smooth. Signed distance fields are not smooth, but fulfil the other requirements, and L_p distances fulfil all the requirements, but are too slow for complex meshes. Convolution filtering fulfills all the requirements and can be controlled by the user.

Figure 4.7 shows transfinite interpolation using Biswas *et al.* (2004) technique used to interpolate properties between two material features, using various signed distances, signed L_p distances and convolution filtering. The two material features (in white and grey stripes on figure 4.7) are set up by the user, two triangles on one side, and an ellipsoid on the other. The bands of colours represent intervals of attribute weights.



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

Figure 4.7: *Transfinite interpolation of material properties between two material features: the top left is the distribution of materials using exact signed distances, top right (b) uses the L_4-dist field. The other examples use filters with various parameters to control the smoothing.*

The weights of the material feature on the left, in this case, have the following values:

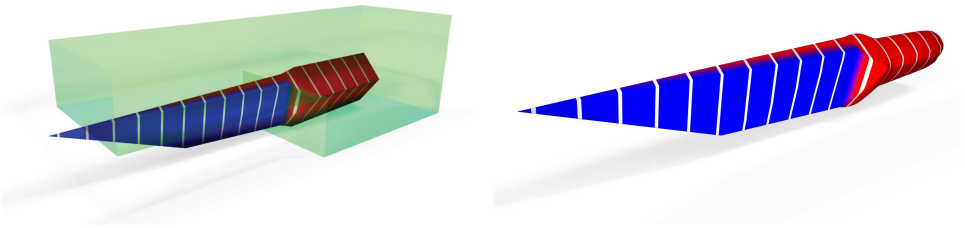
- magenta represents values in the range $[0, \frac{1}{6}]$,
- cyan represents values in the range $[\frac{1}{6}, \frac{1}{3}]$,
- yellow represents values in the range $[\frac{1}{3}, \frac{1}{2}]$,
- blue represents values in the range $[\frac{1}{2}, \frac{2}{3}]$,
- green represents values in the range $[\frac{2}{3}, \frac{5}{6}]$,
- red represents values in the range $[\frac{5}{6}, 1]$.

Using exact distance fields (figure 4.7a), there are visible C^1 discontinuities which are problematic for the solidity of the final object. Using signed L_4-distance field instead of distances (figure 4.7b) is slow, but also poorly approximates the distances far away from the boundary. Figure 4.7 (c) uses convolution filtering applied to both feature shapes with a filter size b of 3.5 reached at capping distance f_c of 5. Figure 4.7(d) uses $b = 1.5$ and $f_c = 5.0$ and figure 4.7(e) uses $b = 1.5$ and $f_c = 2.0$. This figure shows the effect of b over the field as well as the effect of f_c. Figure 4.7 (f) uses $f_c = 0.5$ and $b = 2.0$. Such parameters violate the inequality introduced in equation (3.10). This results in additional zero level sets and an ill-behaved field. Convolution filtering provides results more or less close to the exact distance results, which can be controlled by the user, and the interpolation is smooth (except figure 4.7f).

Overall, the filters succeeded to create smooth material blending. The maximum filter region can be adjusted and the capping distance f_c lets the user control how close to the surface the smoothing should occur.

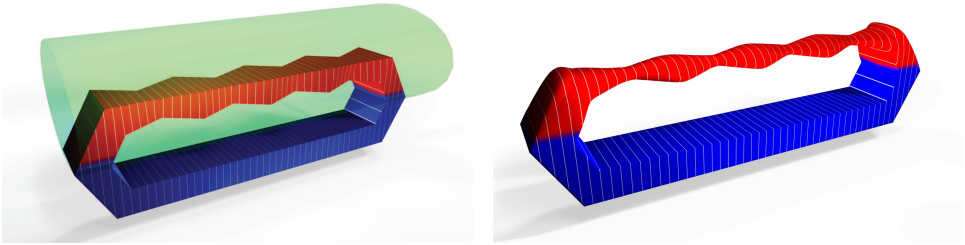
4.2.3 Localized smoothing

Filters can also be used to smooth the shape selectively. To achieve localized smoothing, two C^1-continuous functions are needed. The function f_o represents the original object, while the function f_s represents the smoothing volume. The function h from equation 3.5 is replaced by



(a) (b)

Figure 4.8: *Local smoothing with filtering: (a) shows the original model, and the volume defined to smooth the parts that need smoothing. (b) shows the resulting shape, using smoothing through filters.*



(a) (b)

Figure 4.9: *Local smoothing with filtering: (a) shows the original model, and the volume defined to smooth the parts that need smoothing. (b) shows the resulting shape, using smoothing through filters.*

a function which converts the values from f_s into a filtering size value. Here, a smooth step function remaps the values from f_s into the filtering size. For any point outside the smoothing object, no filtering is required. If the point is within the volume f_s, then the transfer function translates it to the filter size. In this case, convolution filtering can be applied as described in section 3.2.

Figure 4.8 shows how a knife can be made by defining a rough shape with sharp edges all around. The smoothing volume is then defined around the handle and the top of the blade. This smoothing volume controls the size of the filter kernel allowing a smooth handle and a blade sharp only on one side. In the figure, the smoothing amount is indicated by the red surface or the green bounding volume. Figure 4.9 shows a smooth handle can be made from a very basic shape following the same procedure. The smoothing is controlled by the green bounding

volume, and the red surface indicates which part of the object is included in the smoothing volume.

4.2.4 Blending set operations on distance fields

Simple blending set-theoretic operations with addition or subtraction of material for smooth transition between two objects was presented in Pasko *et al.* (1995). The main idea is to apply an R-function defining a set-theoretic operation for two objects and apply locally some displacement. The result of the blending operation is the smooth transition between the initial surfaces. C^1-continuity is crucial for blending operations. The effect of C^1 discontinuities, commonly present in distance fields was shown in figure 3.1. The additive blending shows discontinuities as a sharp edge crossing the otherwise smooth additional material.

In this section, we compare signed distances and the approximate smooth distances introduced in section 3.2 for the purpose of blending unions.

Figure 4.10 shows the effect of C^1 discontinuity on blending union (c) by using signed distances. The crease is visible on figure 4.10d, which is undesirable for a blending union. The figure 4.10e shows the result obtained using convolution filtering, and there is no visible crease in figure 4.10f.

In figure 4.11, a gear model is subtracted from a sphere using a blending difference. The C^1 discontinuities of the signed distance fields create visible creases in the model as seen in figure 4.11b (circled in red). Using convolution filters on signed distance fields, the hard edges are replaced by a smooth surface.

4.2.5 Smooth metamorphosis

Metamorphosis also benefits from C^1-continuous fields. C^1 discontinuities introduce unwanted creases during the transformation. Without

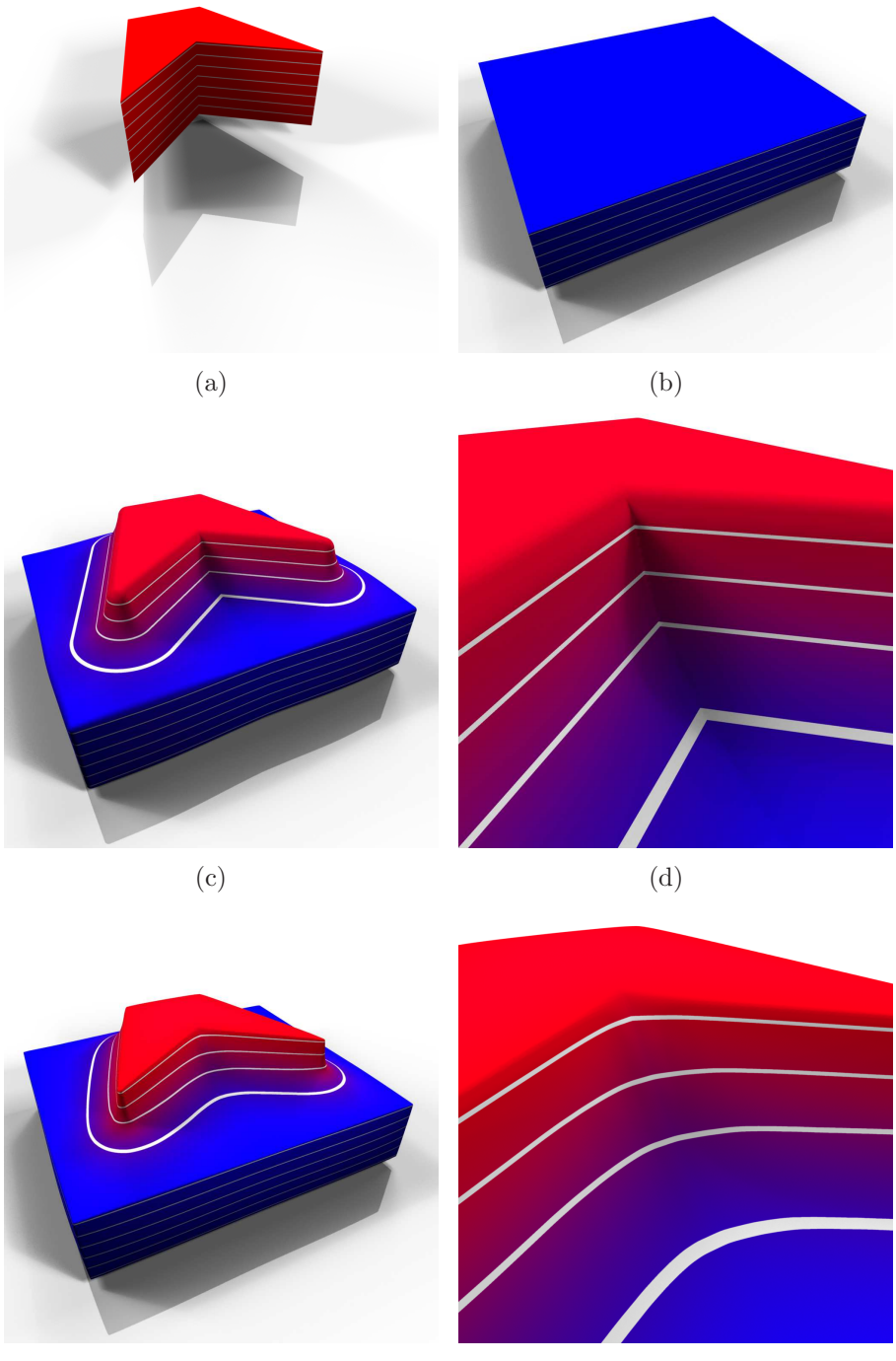
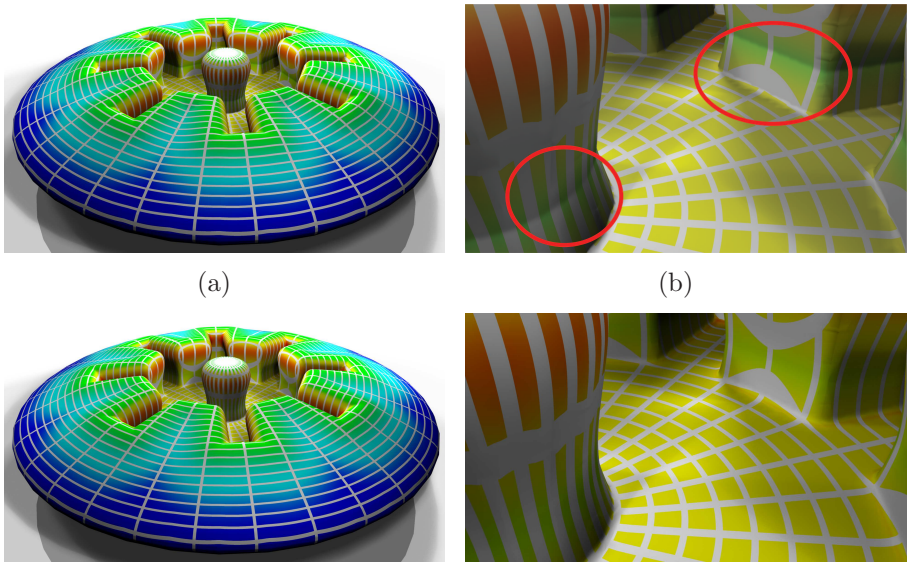


Figure 4.10: *Images (a) and (b) show two initial objects. The image (c) shows the blending union with signed distances and (d) zooms on the discontinuity; (e) and (f) show the same operation with filtered signed distances*



(a) (b)

(c) (d)

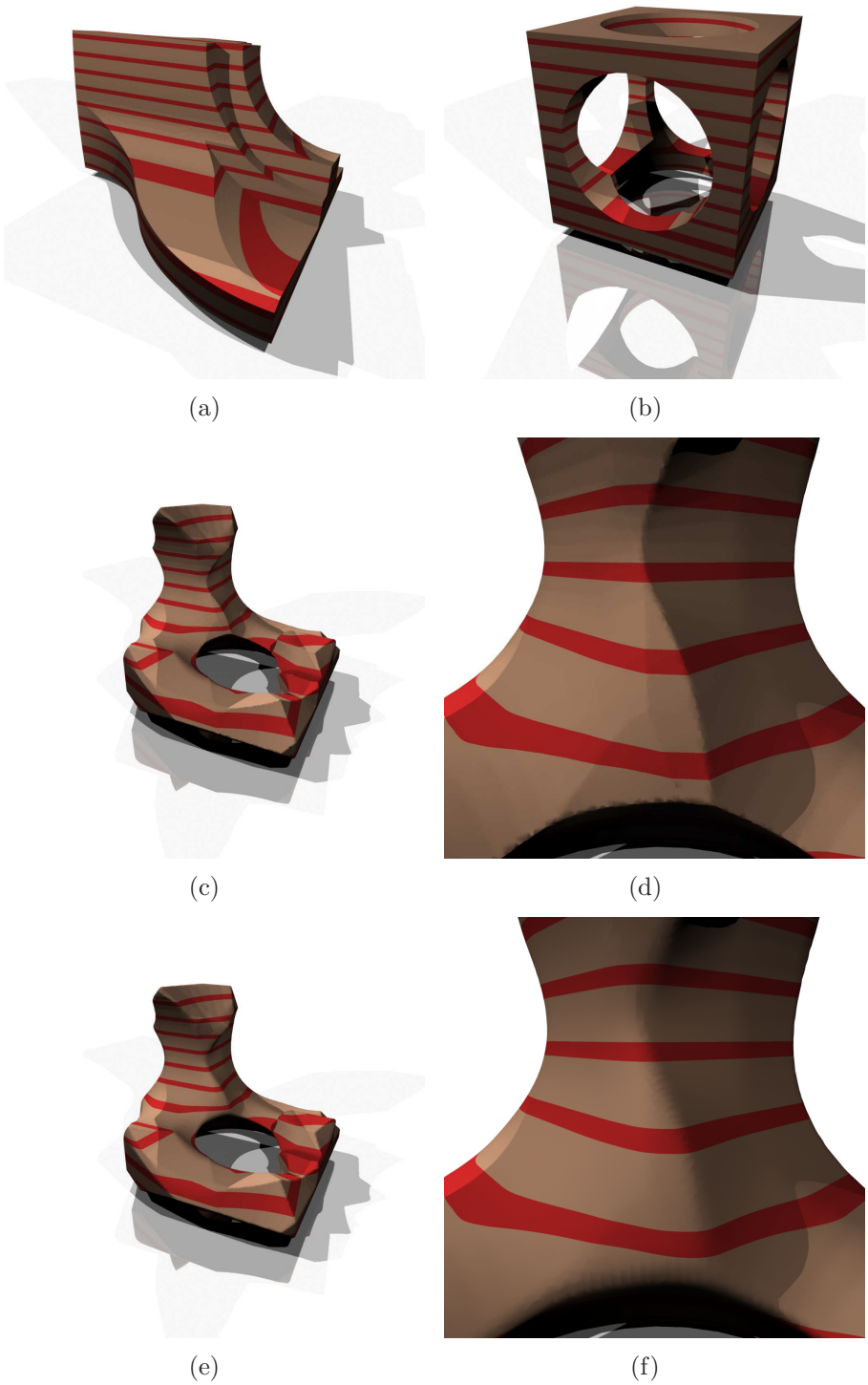
Figure 4.11: *The image a) shows the blended difference between a gear and a sphere with signed distances and b) zooms on the discontinuity; c) and d) show the same operation with filtered signed distances*

such discontinuities, the intermediate shape looks smoother and avoids the grid-like set of creases.

Figure 4.12 shows the metamorphosis applied to a fandisk and a geometric shape with sharp features. The metamorphosis here is achieved using a simple linear interpolation between the values of each field. The metamorphosis with exact signed distances creates creases on the shape surface, sometimes in unexpected places. Using convolution filtering, the surface is smooth, but the source and target shapes of the metamorphosis retain their exact shapes.

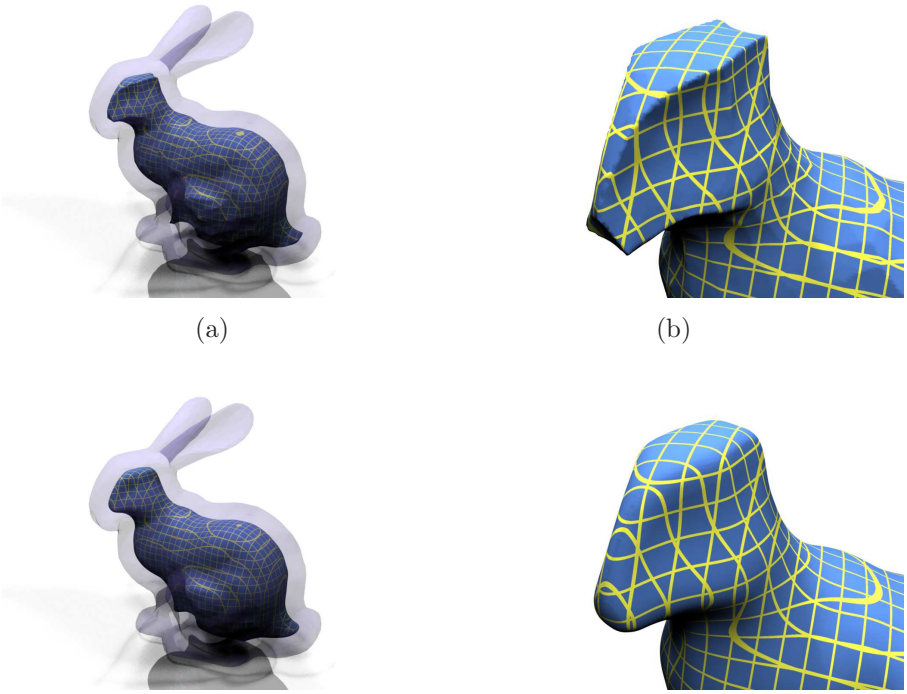
4.2.6 Smooth offset surface

Figure 4.13 shows the offset operator applied to a Stanford bunny using signed distances ((a) and (b)) and using convolution filters ((c) and (d)). Figures 4.13a and 4.13c show the original surface in transparent purple, and the offset surface in opaque blue with yellow lines. Figures 4.13b and 4.13d zoom closer to the head. The convolution filtering solution produces a smooth surface. While an exact offset surface is often required



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

Figure 4.12: *Metamorphosis between a fandisk and a geometric shape with sharp features. (a) and (b) use signed distances. (c) and (d) use convolution filtering.*



(a) (b)

(c) (d)

Figure 4.13: *Offset of the Stanford bunny, (a) and (b) using signed distances, (c) and (d) using our method.*

(using distance fields), a smooth surface allows us to round the sharp corners of the offset, and can find applications in CAD, as an alternative to the exact offset surface. For instance, for 3D printing, objects are often hollow. To create the shell, an inward offset of the surface is used, and then it is subtracted from the original object. A smooth offset would avoid surface discontinuities and improve the robustness of the shell.

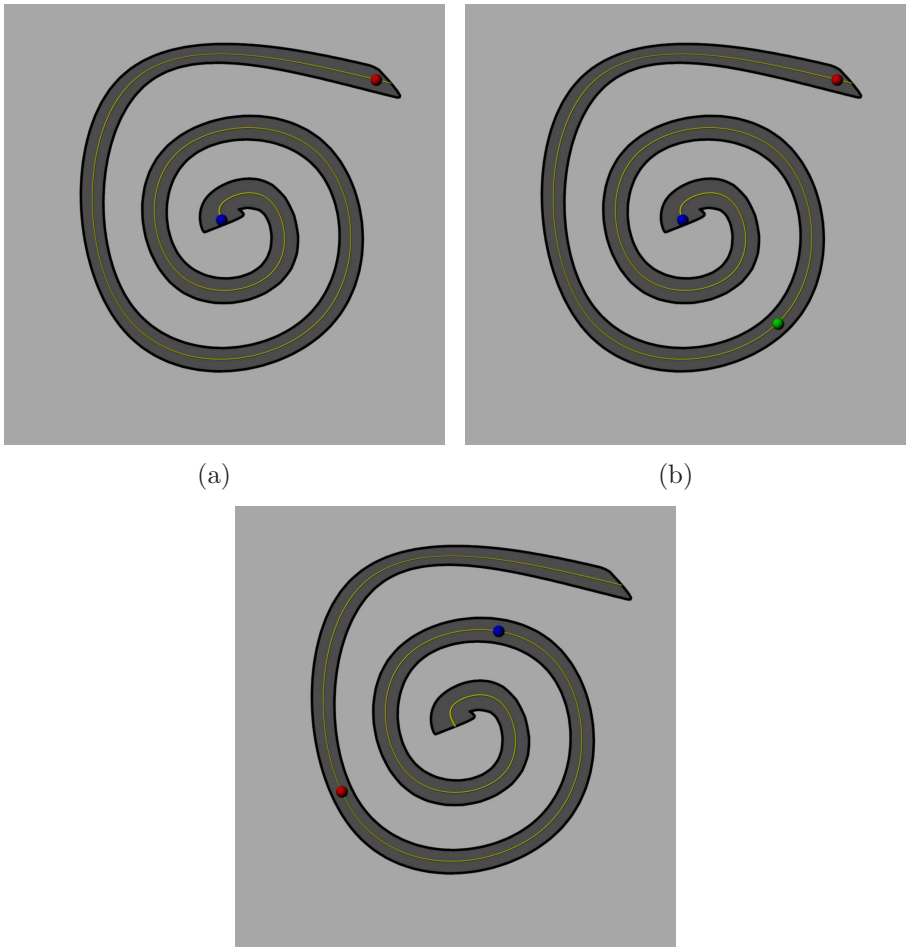
4.3 Shape conforming volumetric interpolation

In this section, we first compare our method presented in section 3.3 with transfinite interpolation as presented in Biswas *et al.* (2004). Then, we show an application of shape conforming volumetric interpolation for microstructure attributes.

4.3.1 Comparison

The comparison is performed visually and numerically. We create a number of cases to show that our method behaves better than the transfinite interpolation as detailed in (Biswas *et al.* 2004). For each case, a number of material features are defined. First, we visually compare the two results. Then, a curve is sampled and a graph of the weights is drawn. The curve might be a medial axis, or in some cases, a curve defined beforehand if the medial axis cannot be represented as a single curve. Monotone behaviour of the obtained function along such a curve means the interpolation is shape conforming, while some oscillations mean that the interpolation is shape independent.

In this section, four models are used, and two or three cases are inspected per model. The first model is a simple 2D spiral, and will show that the problems highlighted in section 3.3 have been solved. The next model is a spring. The spring is chosen because it is a simple model which showcases the issues of Biswas *et al.* (2004) incontestably. Then a



(a) (b)

(c)

Figure 4.14: *The spiral setup cases for comparison*

spanner is used, to show that our method handles more complex shapes than just disk or sphere sweeps. Finally, a rigged character is used to demonstrate complex shapes with complex skeleton, as well as show that the behaviour of our method is consistent with an animation.

4.3.1.1 Spiral

In this subsection, we compare the transfinite interpolation as defined in Biswas *et al.* (2004) with our method. Three test cases are used, and they are shown in figure 4.14.

The first case has two material features, one blue at the centre of the spiral, and one red feature at the end of the central line. Figure 4.15a

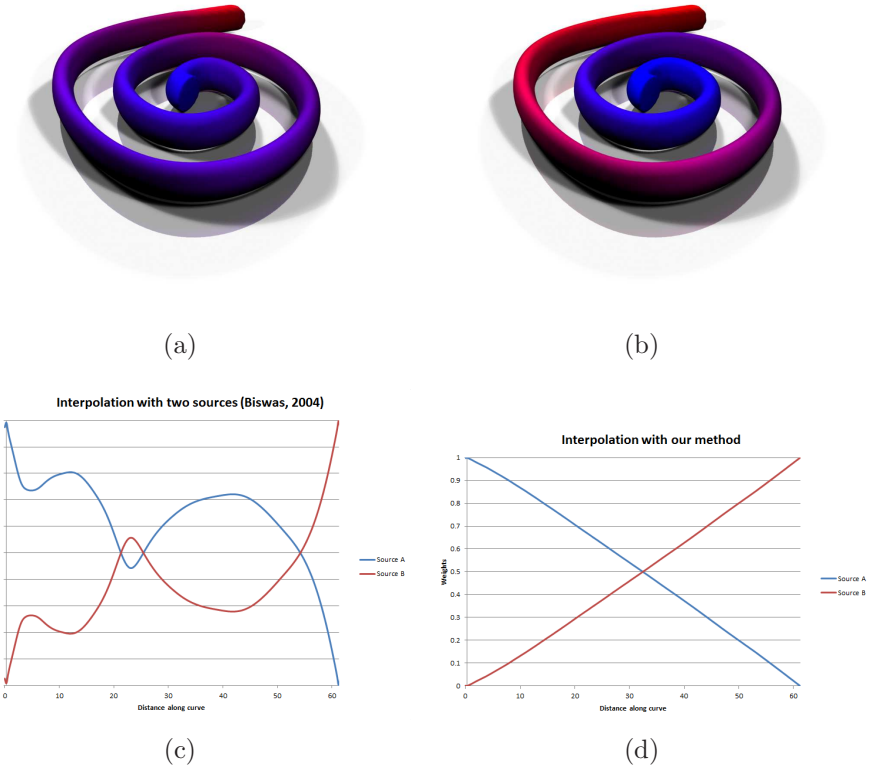


Figure 4.15: *Comparison for the spiral case 1 a) A render of the spiral using Biswas* et al. *(2004), b) A render of the spiral using our method, c,d) the graph of the weights for each source along the medial axis using Biswas* et al. *(2004) (c) and our method (d).*

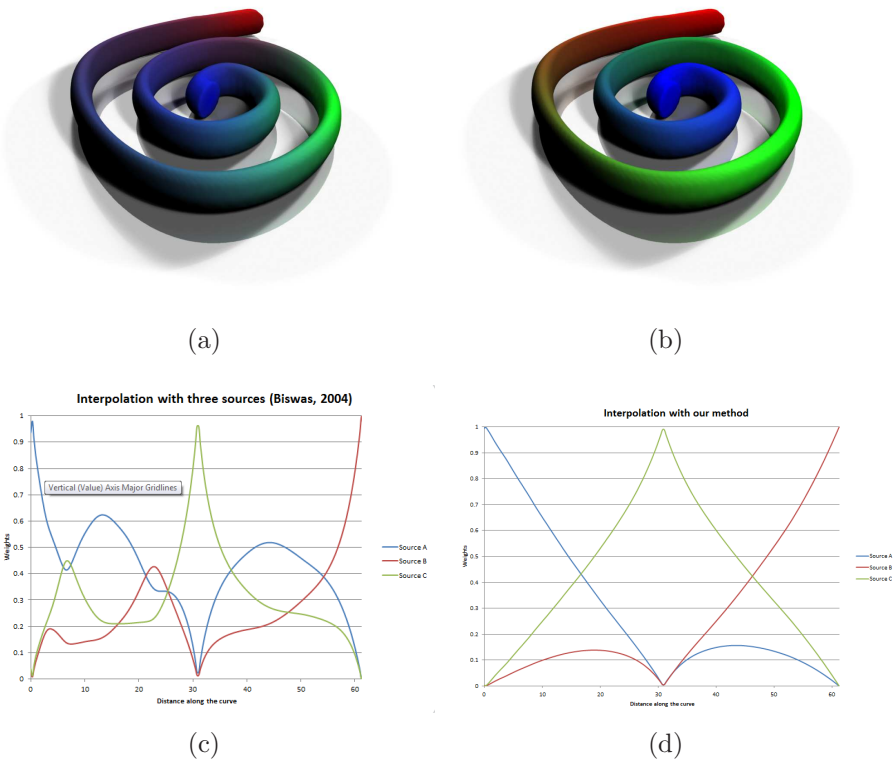


Figure 4.16: *Comparison for the spiral case 2 a) A render of the spiral using Biswas* et al. *(2004), b) A render of the spiral using our method, c,d) the graph of the weights for each source along the medial axis using Biswas* et al. *(2004) (c) and our method (d).*

and 4.15b show the results from Biswas *et al.* (2004) and our method respectively. In figure 4.15a, we can clearly see that the transition is not monotonous as a user would expect. The graphs in figure 4.15c and 4.15d show the weights of each material feature along the medial axis of the spiral. The transfinite interpolation as described in Biswas *et al.* (2004) does not behave as a user would expect. Our method shows a clear, linear interpolation of the properties along the medial axis. This case shows that our technique is shape conformal, and the interpolation depends on the shape being modelled.

The second case shows the behaviour of shape conformal volumetric interpolation with three sources. The material features are defined in

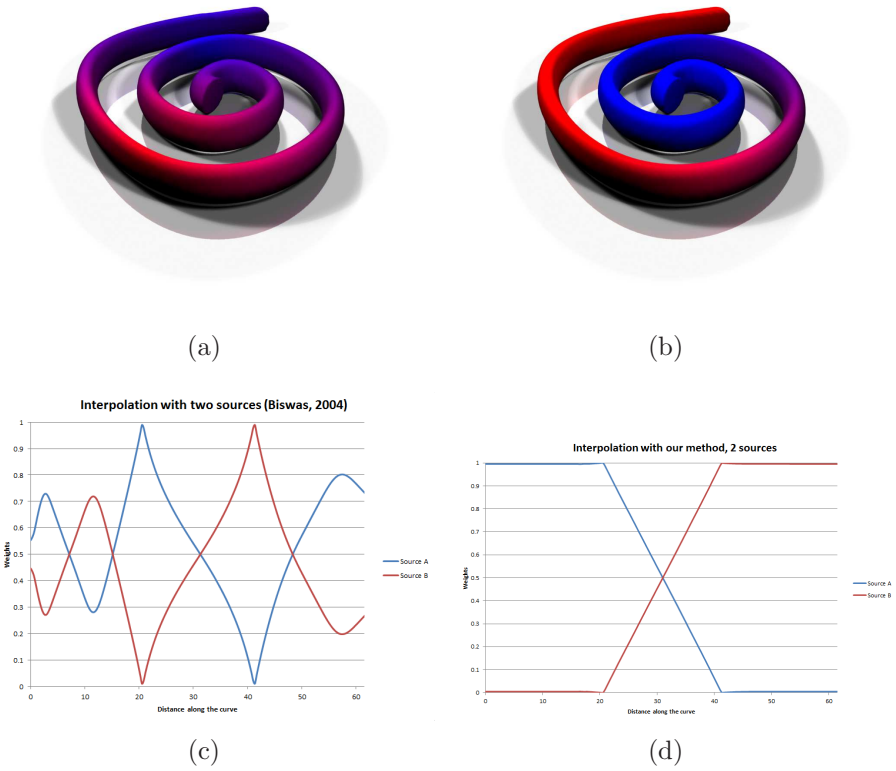
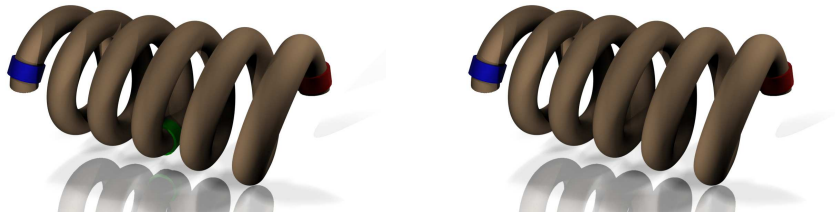


Figure 4.17: *Comparison for the spiral case 3 a) A render of the spiral using Biswas* et al. *(2004), b) A render of the spiral using our method, c,d) the graph of the weights for each source along the medial axis using Biswas* et al. *(2004) (c) and our method (d).*

figure 4.14b, and the results for transfinite interpolation are shown in 4.16a, while the results of shape conformal volumetric interpolation are shown in figure 4.16b. Figure 4.16a shows unexpected behaviour. It does not interpolate smoothly from the first material feature (blue) to the second (green), and then from the second (green) to the third (red). The graph in figure 4.16c shows that the weights for each source are not intuitive along the medial axis. The graph in figure 4.16d shows that our method behaves more intuitively because the interpolation is performed along the medial axis.

The last test case showcases the occlusion of material features in our method. By setting the material features at a third and two thirds



(a) (b)

Figure 4.18: *The spring setup cases for comparison. a) Three material features (blue at the start, green halfway through, red at the end) b) Two material features (blue at the start, red at the end).*

along the medial axis (see figure 4.14c), figure 4.17a shows that past the material features, the interpolation is unintuitive. The weights can oscillate around 0.5 as shown in figure 4.17c. This is true with Euclidean distances. With interior distances alone, it does not oscillate, however it will quickly converge to 0.5 as both distances grow. With our method, points *past* the material features along the medial axis rely only on the closest material feature (see figures figure 4.17b and figure 4.17d).

4.3.1.2 Spring

A spring is used to show the behaviour of our method in 3D. The first case in figure 4.18a has three material sources: at the start of the curve representing the medial axis (blue), one halfway through (green) and one at the end (red). The second case uses only the first and last material features (red and blue).

For the three material feature sources, the undesirable behaviour shown in the spiral test case is exacerbated in 3D. Most weights are oscillating around $\frac{1}{3}$ (see 4.19c), leading to a discoloured spring as seen in figure 4.19a. Shape conforming volumetric interpolation however has an intuitive interpolation which interpolates the material features almost separately, as expected.

Similarly to the spiral case with two sources at the ends of the spiral, our method shows a clear interpolation between the two material features along the curve as shown in figure 4.20d, while the method in

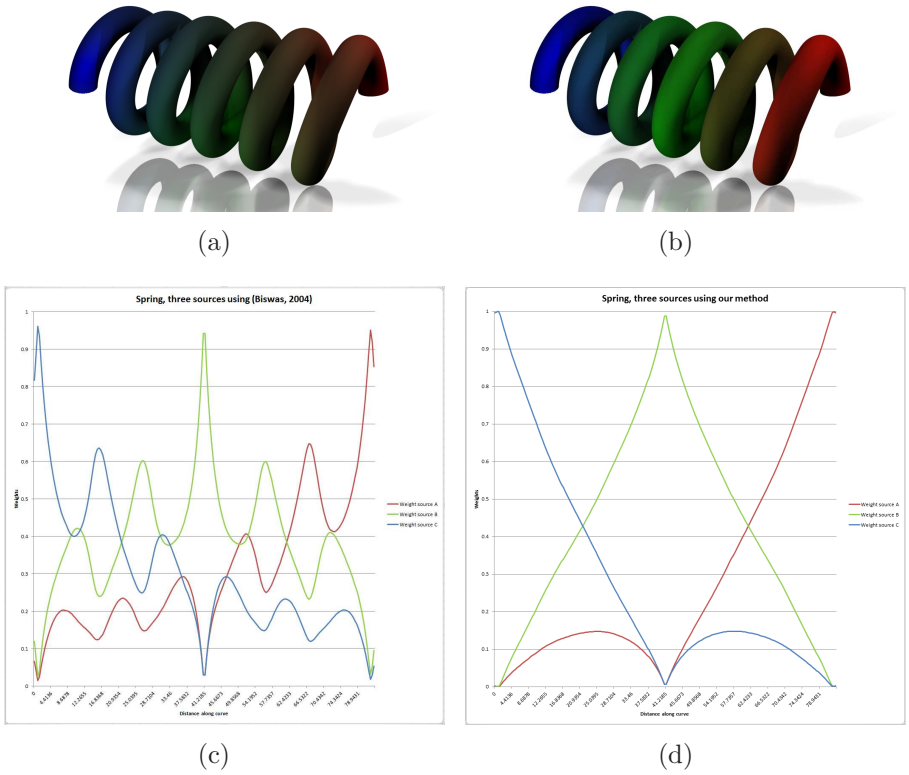


Figure 4.19: *Comparison for the spring case 1 a) A render of the spiral using Biswas* et al. *(2004), b) A render of the spring using our method, c,d) the graph of the weights for each source along the medial axis using Biswas* et al. *(2004) (c) and our method (d).*

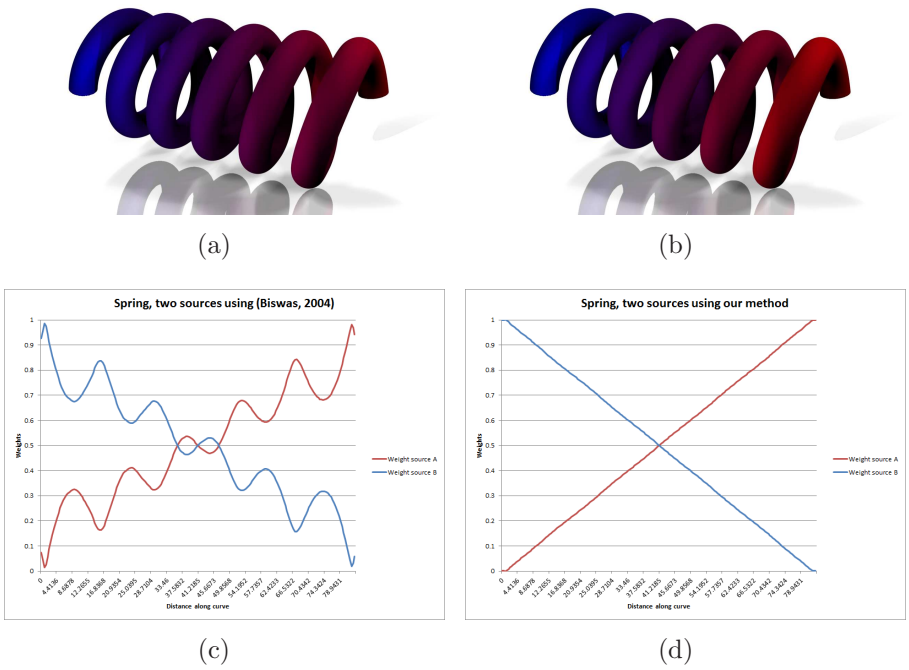


Figure 4.20: *Comparison for the spring case 2 a) A render of the spiral using Biswas* et al. *(2004), b) A render of the spiral using our method, c,d) the graph of the weights for each source along the medial axis using Biswas* et al. *(2004) (c) and our method (d).*

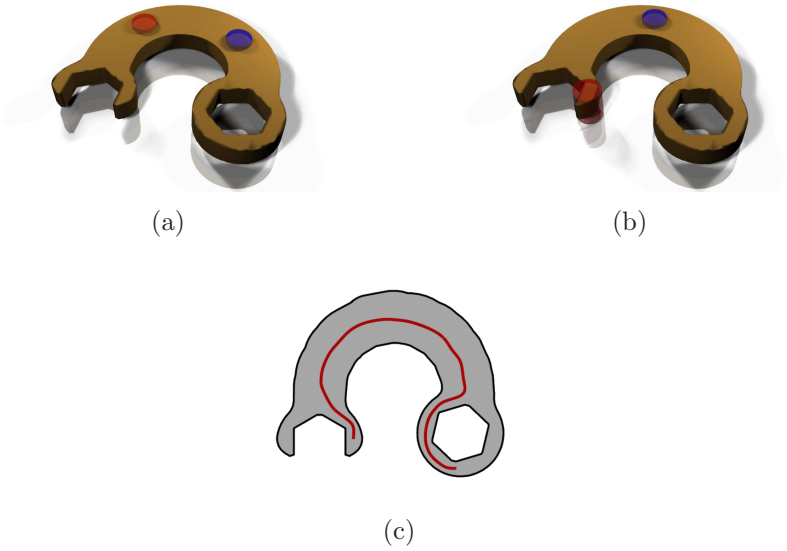


Figure 4.21: *The spanner setup cases for comparison , a) first case, b) second case, c) the curve used for evaluation of the weights*

Biswas *et al.* (2004) fails to provide a monotonous interpolation along the curve as shown in figure 4.20c.

4.3.1.3 Spanner

We define two more test cases on a bent spanner model. The material sources for the first case is shown in figure 4.21a, and the second case in figure 4.21b. Since the medial axis is not simply a curve, we selected a curve which is partially on the medial axis. A user would expect a monotonous interpolation along this curve. The curve is shown in red in figure 4.21c.

Figure 4.22a compares our results with those of Biswas *et al.* (2004). The results are visible in figure 4.22a and 4.22b. The graph of the weights along the curve shown in figure 4.21c are shown in figure 4.22c and 4.22d. The interpolations in figure 4.22a and 4.22b are identical in between the two material features, which is what we expect, and past the material features, our method retains the values of the closest material feature, which is the desired behaviour.

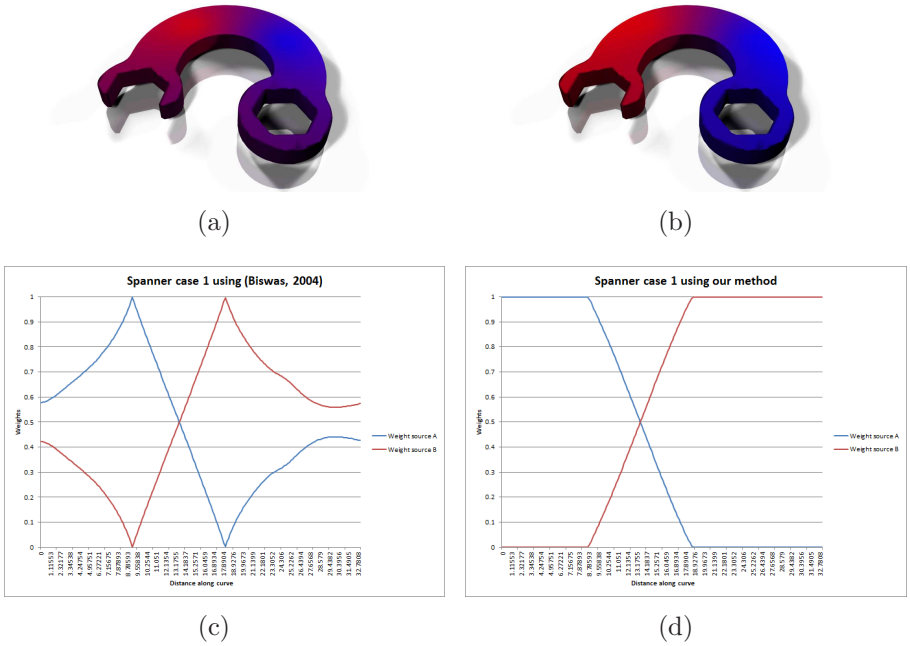


Figure 4.22: *Comparison for the spanner case 1 a) A render of the spiral using Biswas* et al. *(2004), b) A render of the spanner using our method, c,d) the graph of the weights for each source along the curve shown in 4.21c using Biswas* et al. *(2004) (c) and our method (d).*

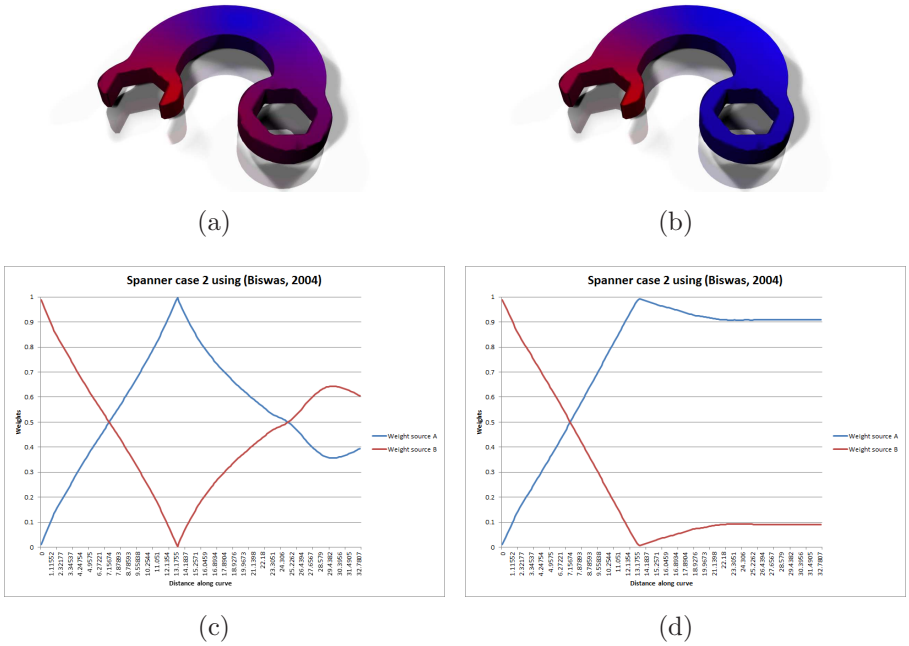


Figure 4.23: *Comparison for the spanner case 2 a) A render of the spiral using Biswas* et al. *(2004), b) A render of the spanner using our method, c,d) the graph of the weights for each source along the curve shown in 4.21c using Biswas* et al. *(2004) (c) and our method (d).*

(a) (b) (c)

Figure 4.24: *The three different poses used for comparison with the material features. Red feature on the right hand, blue feature on the left hand, yellow feature on the right foot, cyan feature on the left foot. Charlie character and its rig courtesy of Jahirul Amin and the NCCA.*

In this last case, we show that Biswas *et al.* (2004) can be unintuitive in simple cases. The results from Biswas *et al.* (2004) in figure 4.23a are unsatisfactory. The graph in figure 4.23c shows that the weights are not conformal to the shape. Our method (figure 4.23b) provides a more intuitive interpolation since the interpolation happens in-between the material features.

4.3.1.4 Character

The shape conformity of our method can be shown by using a character in different poses. In this example, a character is used with material features at the hands and feet. The interpolation should be consistent when the model is animated or modified slightly, given that the topology does not change. For a rigged model, if only the pose is changed, then a user would expect similar values along the skeleton or on the surface. In this subsection, we verify this using three poses of the Charlie character. We start with a visual comparison, and then look at the weights along several curves.

The figure 4.24 shows the material features used with the character Charlie for each pose:

- The right hand feature is in red
- The left hand feature is in blue
- The right foot feature is in yellow

(a) (b) (c)

Figure 4.25: *Transfinite interpolation using the material features defined in figure 4.24*

(a) (b) (c)

Figure 4.26: *Our method using the material features defined in figure 4.24*

- The left foot feature is in cyan

Figure 4.25 shows the results using transfinite interpolation as shown in Biswas *et al.* (2004). There are a number of counter intuitive behaviours. In figure 4.25a, the legs are mostly green except at the features. A user may expect the right leg to be mostly yellow, and the right leg mostly cyan. In figure 4.25b we notice that the hands have an influence on the legs and hips. The left hip is mostly blue, and the legs are now almost purple. Figure 4.25c has the same issues.

Figure 4.26 shows the results using our method. The colours of the body parts are consistent across the various poses, and the colours of the legs are more intuitive.

We confirm the visual comparison with graphs of the weights along several curves. For each pose, three curves are defined. One curve connect the right hand to the left hand (arms curve). One curve connects the right foot with the left foot (legs curve), and finally, one curve connects the neck with the lower back (spine curve). Figure 4.27 shows the weights generated by transfinite interpolation along each curve. Each row corresponds to a curve (arms, legs, spine), and each column corre-

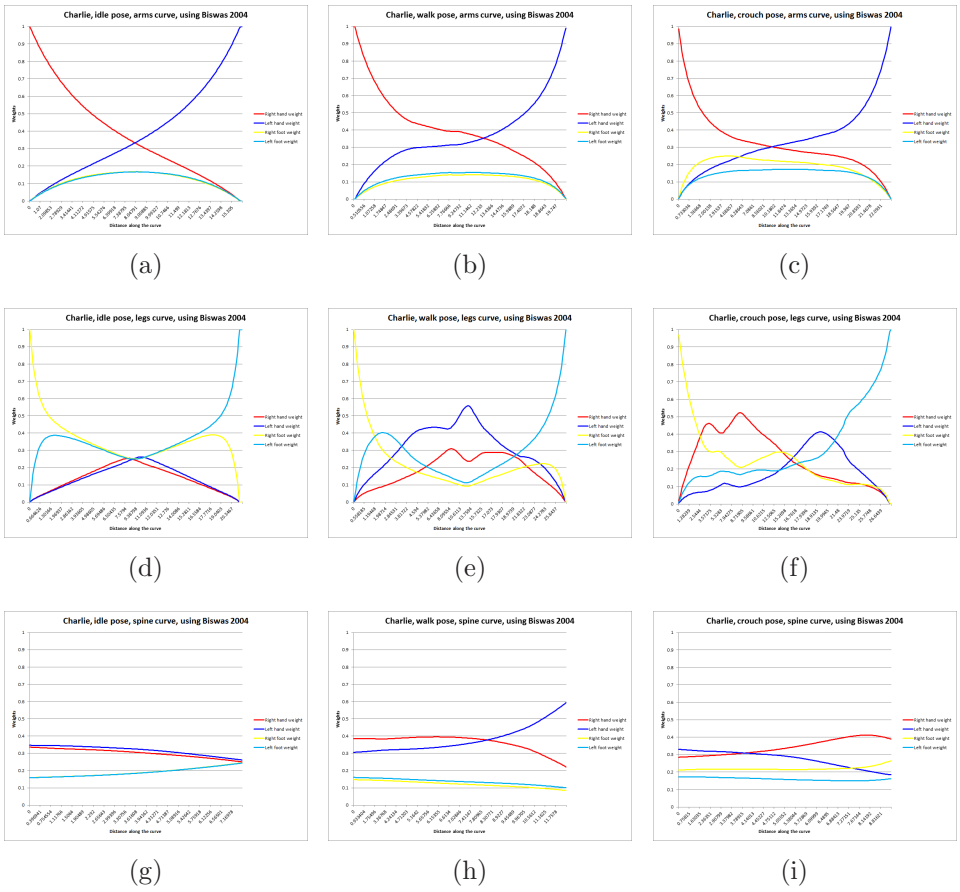


Figure 4.27: *Graphs of the weights along the curves using Biswas* et al. *(2004). First row for the arms curve, second row for legs curve, last row for the spine curve. Each column corresponds to a pose (idle, walk, crouch)*

sponds to a pose of the character (idle, walk, crouch).

We notice that the weights along the arms curve are not consistent across the different poses (figures 4.27a, 4.27b, 4.27c).

The legs curve is even more inconsistent, and can even become incoherent. The weights along the legs curve in the idle pose (figure 4.27d) show that the right foot and left foot quickly reach similar weights. The weights for the other two poses (figure 4.27e and figure 4.27f) have oscillating weights which do not resemble each other. This shows that the behaviour is counter intuitive and inconsistent.

Finally, the weights of the spine curve shown in the third row also show a lot of variation. The figure 4.27g (idle position) follows the expectations with higher weights of the hand material features towards the neck, and almost equal weights for all the material features at the lower back. For the other two poses (figure 4.27h and figure 4.27i), the graphs are very different, further confirming that it does not behave intuitively.

Figure 4.28 show the weights along the same curves and same poses using our method. The first row shows that the weights along the arms curve are consistent across different poses. For the legs curve (figures 4.28d, 4.28e and 4.28f), the curves are marginally different. Finally, for the spine, the curves have noticeable differences. This is due to the body bending and blending with the arms and legs changing the interior distances and the Voronoi diagram. However, this remains acceptable, since there are no oscillations, and the curves still follow a similar trend.

4.3.2 Shape conforming interpolation of microstructures

In figure 4.29, shape conformal volumetric interpolation is used to control microstructure parameters across a spanner. More parameters can be defined for each material source and then interpolated. The function driven foam model used in this example is described in appendix C.

Thus, in figure 4.29a, different lattice structures are defined for dif-

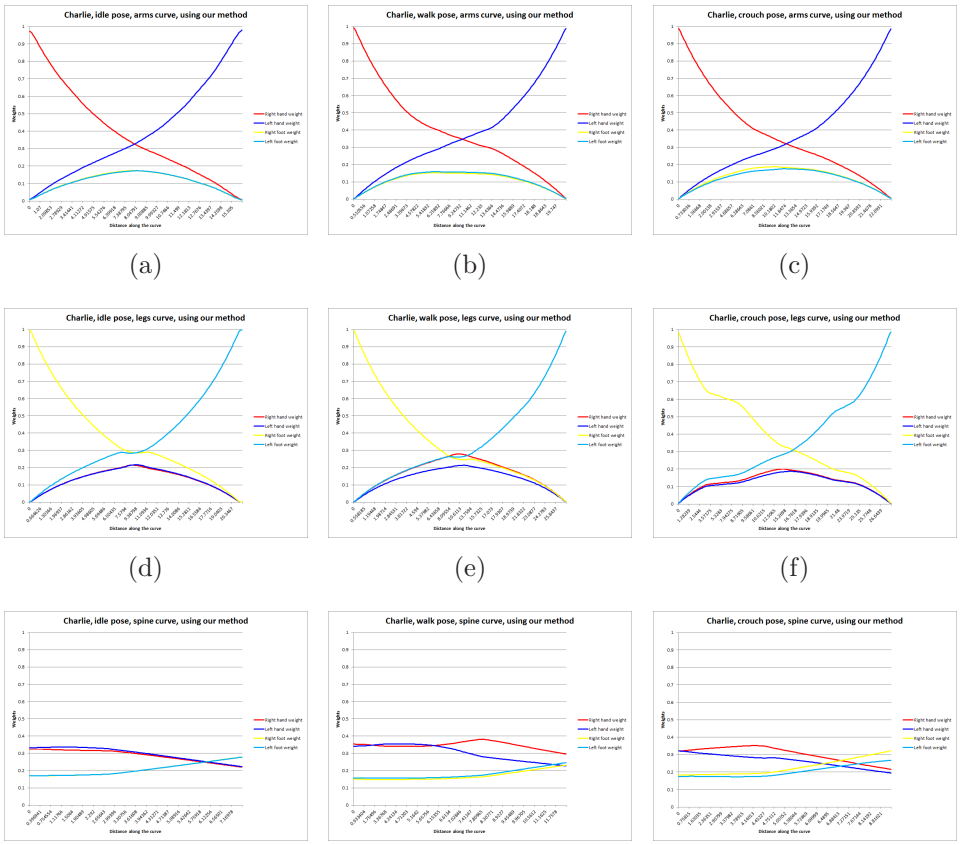


Figure 4.28: *Graphs of the weights along the curves using our method. First row for the arms curve, second row for legs curve, last row for the spine curve. Each column corresponds to a pose (idle, walk, crouch)*

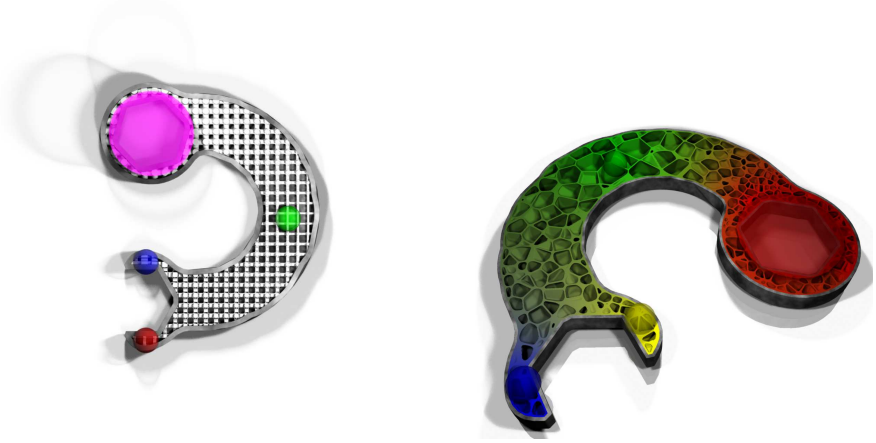


Figure 4.29: *Controlling microstructure properties: a) Various regular lattices interpolated, c) Foam structure with several parameters controlled by user features.*

ferent material sources: a cylindrical lattice with different thickness for two material sources (large for the magenta feature, thin for the green feature) and a cubical lattice with different thickness for another two material sources (large for the blue feature, thin for the red feature). The interpolation provides a weighting for each feature, and thus, blends all the lattices together using a weighted sum of the function values of each lattice. Finally, in figure 4.29b, four features have several values set up. The objective here is to control the attributes of the foam structure which are the average cell size, the wall thickness and the roundness (blending amount). Each feature provides a value for each of those parameters. The large red feature uses a small cell size, a large blending value and thick walls. The green feature contrasts with narrow walls, practically no blending and ample cell size. The last two features (yellow and blue) have high blending, thick walls. The yellow feature has the same cell size as the red feature while the blue is slightly larger.

4.4 Space time transfinite interpolation

In this section, three applications of space time transfinite interpolation for volumetric interpolation are shown. As shown in section 3.4.3, there are no other methods which can fulfil the requirements defined in section 3.4.2. The advantages of the proposed approach in section 3.4 are:

- Arbitrary volumetric properties can be interpolated through time
- The interpolation is driven by the source and target shapes (see figure 3.24)
- Function value at any point in time can be obtained by evaluating a simple closed form expression
- It is easy to use and control (see section 3.4.7 or section 4.1.3)
- Small scale details are interpolated thanks to the exact function representation

4.4.1 Space time interpolation of material properties

Constant colours are not very useful in the definition of realistically looking objects. However, in order to interpolate material and other attributes distribution, a volumetric representation of such attributes is needed. To do so, the following different approaches (procedural, space enumeration and surface texture extrapolation) are usually employed.

- Solid texturing can be used to define the material properties or any tri-variate function. They have been successfully used to describe marble, wood, and many other natural heterogeneous materials (see the "watermelon" model in figure 3.22). While those methods are the most accurate and explicit methods, they are unpopular because they often require knowledge of scripting or other technical skills.

- Space enumeration is defined by a constant colour for a given space partition (see the "cartoon cat" model in figure 3.22). Note that space enumeration can be combined with solid texturing. Some particular care has to be taken for points which belong to several partitions or to none at all. In the case of a point belonging to several partitions, a common approach is to give a higher priority to a particular partition over the others. If the point does not belong to any partition, then a default colour is used, such as white or grey.

- Surface textures are popular in creating heterogeneous looking surface colours and properties. But they only define properties at the surface. A volumetric representation of those surface textures can be achieved using the closest point of the mesh. This is neither an accurate nor a continuous representation, but gives reasonable results especially during an animation sequence. The "banana" model in figure 3.22 uses this technique. However, to achieve better results, alternative techniques can be used such as the mean value coordinates (Ju *et al.* 2005). Unfortunately, mean value co-

Figure 4.30: *The animation from a Buddha to a teapot, with a box cut out to see the material inside the intermediate shapes. The volumetric properties (colours) are interpolated using space time transfinite interpolation.*

ordinates are time consuming and often impractical for real meshes. This approach is not suitable for interactive applications but gives smoother and more intuitive results.

Figure 4.30 shows a few frames from an animation generated in Maya. A segment of the shape is deliberately removed on every frame so that the internal distribution of colours becomes visible. The material properties inside the intermediate shapes show that locations where the head of the Buddha figurine used to be still retain the Buddha's material properties mostly, while the inside of the teapot is a more even blend of both materials. The geometry for the metamorphosis of figure 4.30 was created using space time blending (Pasko *et al.* 2004). Space time transfinite interpolation provides the colour information at any point in space and time. The colour information for the Stanford Buddha in figure 4.30 comes from the surface texture, while the Utah teapot colour information comes from solid texturing. To produce this example, we used our Maya plug-in described in section 4.1.

4.4.2 Space time interpolation of displacement and other attributes

Space time transfinite interpolation can also be used for other properties such as displacement. Displacement is an important visual clue that help us understand the material properties of an object, just like pigmentation (colours). The displacement is applied to the intermediate shapes

Figure 4.31: *The Stanford bunny is transformed into the Stanford dragon, intermediate colours and displacement are produced by the space time transfinite interpolation*

following the basic shape formation. The displacement information is available at any point in space, but existent renderers will only evaluate it at the surface. This is purely cosmetic in nature and the interior of the shape remains completely solid.

The example in figure 4.31 defined two displacement values through solid texturing. The two attributes were then blended together using equations 3.25 and 3.26. The displacement information is a single value in the range $[0, 1]$. The space time transfinite interpolation provides this value during the animation and provides at render time, a single displacement per surface sample which can be used to offset the surface along its normal. The space time transfinite interpolation is equally applicable to any attribute which can be linearly interpolated, such as roughness, opacity and reflectivity.

4.4.3 Space time interpolation of microstructures

Microstructures are internal spatial geometric structures with size of detail orders of magnitude smaller than the overall size of the object. Microstructures can be described by a function with several parameters such as orientation and thickness. As any pointwise attribute, these parameters can be interpolated with a space time transfinite interpolation. Space time transfinite interpolation can be used as a solution to control the microstructure distributions when a union of two objects with two different microstructures is performed.

In figure 4.32, two types of microstructures are defined. Microstruc-

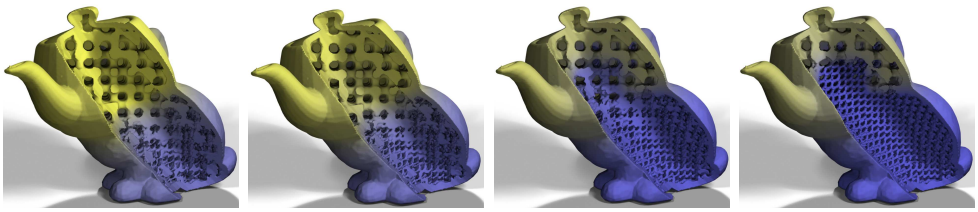


Figure 4.32: *The microstructure parameters are interpolated across the volume. Here the parameter is the mix amount between the lattice microstructure and the interlaced thread microstructure.*

ture A is a regular lattice represented by the function m_a, and microstructure B, which is made up of interlaced threads, is represented by the function m_b. The parameter interpolated is the mix value ω, set to 0 for the teapot, and 1 for the bunny. The final microstructure function is defined as follows:

$$m_f = m_a \cdot (1 - \omega) + m_b \cdot \omega$$

The blending union operation is applied to the objects, and the result is filled with the microstructure represented by m_f. The time parameter is used as a designer tool to help determine how much each of the source objects should influence the resulting internal microstructure.

4.5 Morphological shape generation

In this section, the application results of the morphological shape generation introduced in section 3.5 are presented.

The metamorphosis formulation described in section 3.5 was successful applied to several projects (see section 4.5.2 and section 4.5.3). In figure 4.33a, seven virtual chairs were mixed and digitally materialized. Figure 4.33a shows a small selection from thousands of models generated, the three chairs in pink were created for testing. The designer provides a number of chairs and defines their corresponding features. While any features and correlations can be defined (and redefined), here the designer used four features for the legs, two for the arm rests, one for the seat and one for the back. The deformation and metamorphosis values

(a)

(b)

Figure 4.33: *Results of the user-controlled group metamorphosis: a) the seven initial chairs (blue) with user-defined and weighted features result in the newly generated shapes (pink). b) The miniature chairs 3D printed using an Envision TEC Ultra 3d printer.*

are provided as two $k \times n$ matrices where k is the number of features, and n the number of models. While internally the sum of the weights for a feature must equate to one, the designer can use arbitrary weighted values, since the weights are normalized internally. Additionally, to ensure the objects are printable, feature based offsets are provided per feature. Some examples have been fabricated using an Envision TEC Ultra 3D printer (see figure 4.33b). The printing process is straight forward with FRep. Most printers require a mesh as an STL file, therefore marching cubes (Lorensen and Cline 1987) can be used to generate a manifold mesh. If an image stack is required, then simply sampling the function is sufficient.

This method was successfully employed in two independent projects by an artist and a designer, without supervision. The resulting shapes have been 3D printed.

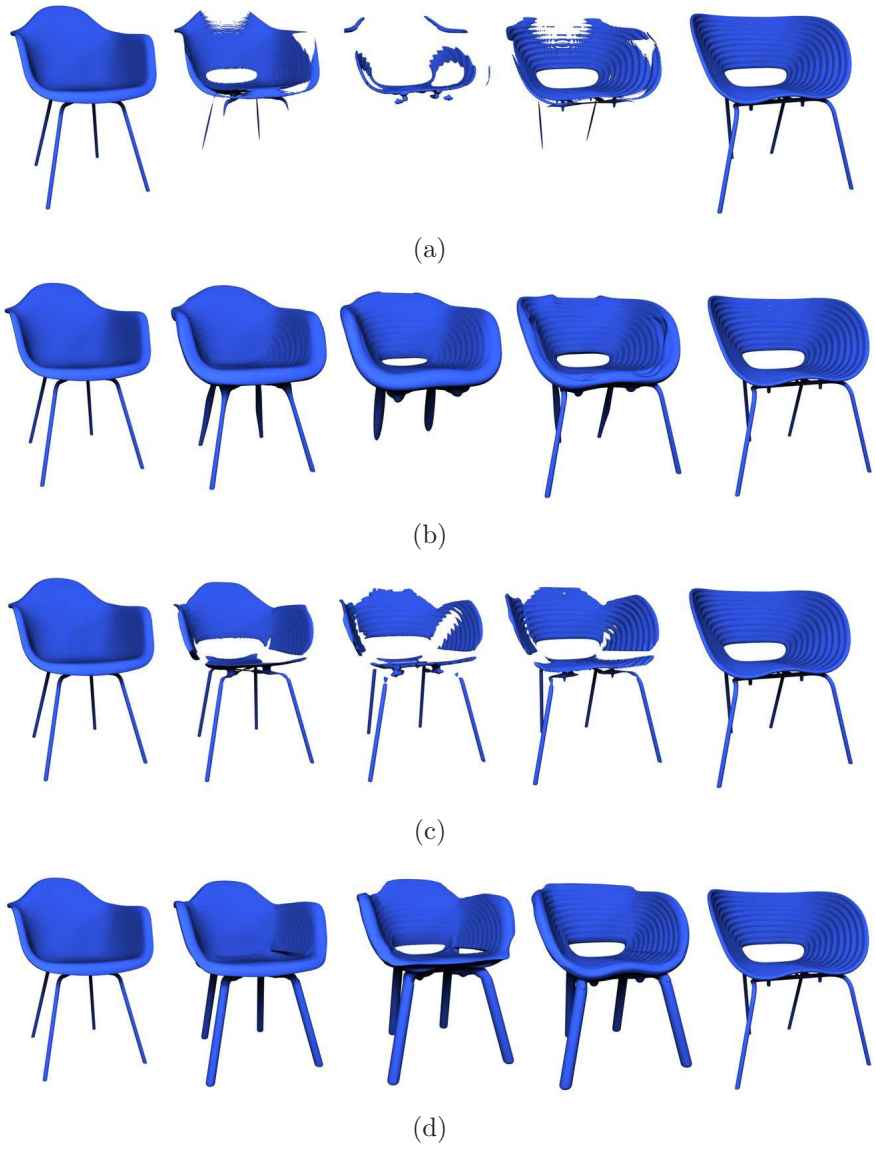
4.5.1 Comparison

The objective of this project was to apply metamorphosis to a set of chairs. User control morphological shape generation allows us to obtain far better results than automatic metamorphosis (see figure 4.34) even where the user can define feature elements. In the example of the models with thin feature elements, linear metamorphosis does not produce sensible results while using space-time blending leads to some features disappearing and growing in unwanted places. Feature-based metamorphosis Lerios *et al.* (1995) produces better results as a designer can choose the best values for each feature set and prevent, for example, thin or small features from vanishing in various stages of the metamorphosis. Our method produces better chairs which are printable, and is generalized to group metamorphosis, and per-feature interpolation. Figure 4.34d shows the pairwise metamorphosis introduced in section 3.5.5. In this example, the chairs can be printed and will be functional since the legs are connected to the seat, unlike the example in figure 4.34c. In figures figure 4.34a and figure 4.34b, the legs completely disappear. Since the offsets are controlled by the user, the amount of extra material is up to the user, who could make the legs very thin (which might be a problem for printing, but not for visualization), or large like in the example, and safe for printing.

4.5.2 Chairgenics

It should be noted that the impetus for this research has been an ongoing collaboration with a well-known designer to develop a practical and usable system for generalized morphological design. Of particular interest was applying this method to morph between famous and well known chair designs. The goal was to *"breed"* classic seating forms in a quest to find the most beautiful, most perfect chair "species". Figure 4.35a shows a sketch of the proposed exhibit in a contemporary art museum, and a number of chairs printed in miniature in figure 4.35b.

Jan Habraken, the designer, was unable to realize his vision of breed-



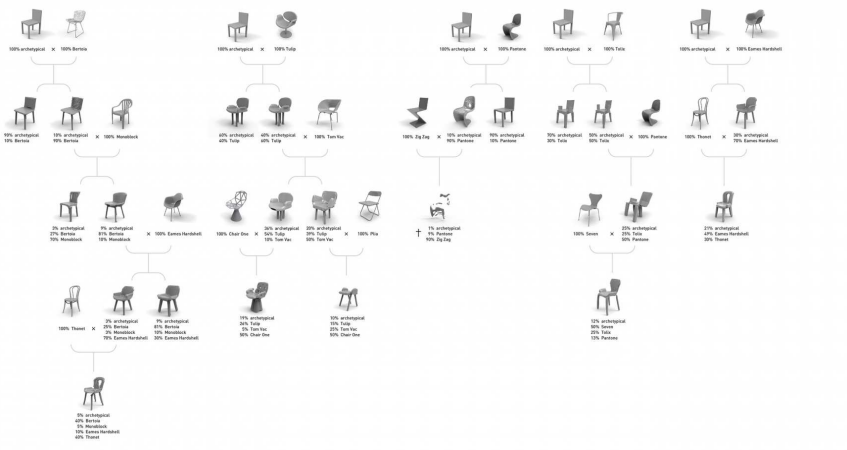
(a)

(b)

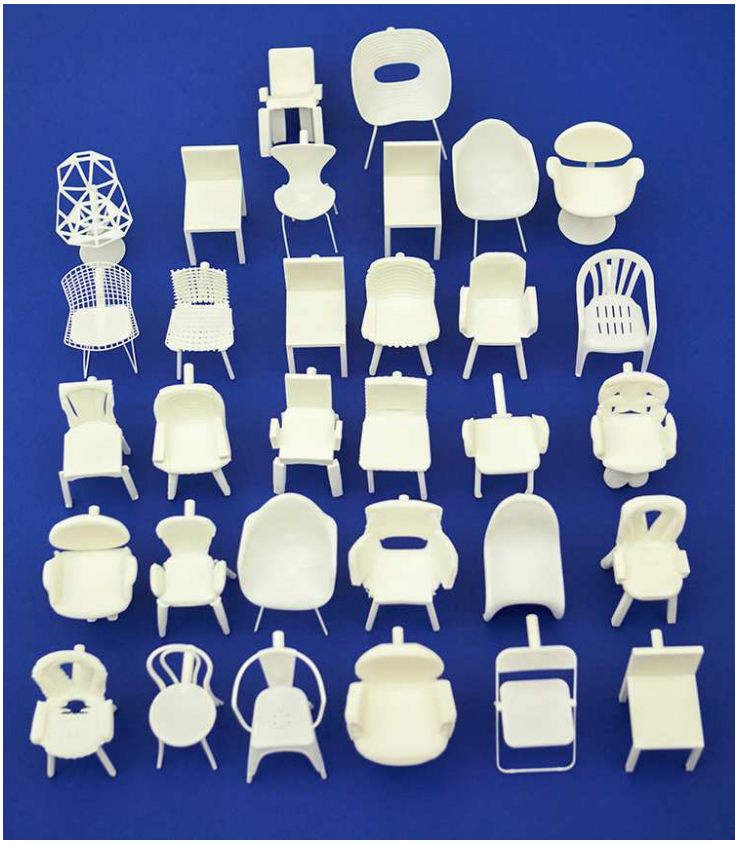
(c)

(d)

Figure 4.34: *Comparison of different metamorphosis methods for objects represented with signed distance fields. a) linear metamorphosis, b) space-time blending, c) feature-based volume metamorphosis, d) the pairwise metamorphosis with space-time blending and additional local offsets.*



(a)



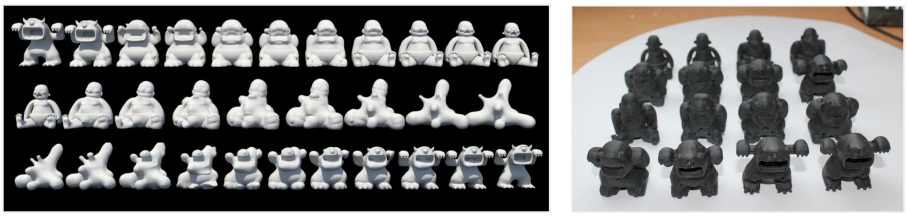
(b)

Figure 4.35: *a) A sketch presenting the concept of the chairgenics project by Jan Habraken, b) the original chairs and the ones generated by the design printed, images courtesy of Formnation.*

Figure 4.36: *The chairs created by the designer Jan Habraken, images courtesy of Formnation and Jan Habraken*

ing chairs for several years due to the limitations placed on software tools by B-Rep and mesh representations. In using and playing with the developed tool, he was for the first time able to truly define feature relationships and breed between any target chairs or objects he desired in a fully automated fashion. As in any breeding process some of the resulting outcomes were undesirable or even non-functional, however the power and capability of the process allowed a fast and broad exploration of the design space with a large number of iterations and resulting chairs, that could be sent to 3D printing directly, without a difficult and tedious mesh repair process.

The resulting chairs were 3D printed at miniature scale along with the ideal chair, selected by the designer through a "breeding process", being printed full scale. The designer's selection is shown in figure 4.36. All the chairs and models were exhibited as the physical artifacts of the synthetic breeding process (Hirst 2013; Stinson 2013; Voyatzis 2013) and the work has been included in Labaco (2013) along with the exhibition



(a) (b)

Figure 4.37: *A Zoetrope with 3D printed objects, created by William Copley. a) the virtual shapes, created by our plug-in and b) the printed objects.*

(a)

Figure 4.38: *A Zoetrope with 3D printed objects, created by William Copley. a) the virtual shapes, created by our plug-in and b) the printed objects.*

"Out of hand: Materializing the Postdigital".

4.5.3 Printed 3D Zoetrope

William Copley, a student at Bournemouth University, used the Maya plug-in to successfully create a physical Zoetrope which show the transformation between a baby and a monster (Copley 2014) using 3D printed objects rather than 2D figures. The shapes generated by our plug-in are shown in figure 4.37a, and the printed shapes in figure 4.37b. Note that these objects were printed on a low quality printer, which leaves many artifacts otherwise not visible in Maya. The final prototype of the Zoetrope is visible in figure 4.38.

4.6 Summary

In this chapter, many applications were shown to illustrate the techniques introduced in chapter 3. The proposed convolution filtering was evaluated and various applications of convolution filtering were investigated, notably using a transfinite interpolation to provide a smooth material transition based on the boundary representation of material features. Shape conforming volumetric interpolation was compared to transfinite interpolation, and was used to control material distributions and the attributes of a microstructure. The interpolation field created complex internal structures adapted to the shape of the object. Then, the space time transfinite interpolation was illustrated with several attributes including microstructures and surface displacement. Finally, a concrete problem was solved using distance fields and the method described in section 3.5 to generate morphological shapes from various source objects.

Chapter 5

Conclusion

First of all, the topics covered in this thesis are summarized below.

In chapter 1, an overview of the subject area was presented. The overview includes the characterisation of the current state of shape modelling, and, in particular, heterogeneous volume modelling. The overview also covered some of the issues of classical modelling tools and showed why better tools is important.

In chapter 2, an investigation into works related to distance based heterogeneous volume modelling was presented. This is subdivided into four parts. First, the various object representations and their limitations were introduced. While BReps are important, FReps seemed more relevant to heterogeneous volume modelling. Then, heterogeneous volume modelling was presented in more detail. This included several solutions with various object representations, but the focus was on scalar fields. Distance fields were introduced, including definitions, current solutions, and their applications. Finally, the background chapter closes on a few geometric operations which are important for the rest of this thesis.

In chapter 3, the general approach of this work, which relies on heterogeneous attributes and properties as functions of time and distances, was introduced. This framework led to a set of problems to be solved. The first problem was smooth signed distance fields, where the exact surface is maintained, but the rest of the field is smoothed out. Then,

an improvement of *feature*-based volumetric interpolation was proposed that provided more intuitive volumetric interpolation in-between material features. Then, smooth distance fields were used to interpolate between two volumetric attribute distributions across time as the shape of the object was changing. Finally, a solution for mixing various parts of various objects to create new shapes was introduced in the last section of that chapter.

In chapter 4, the experimental results and applications of the methods introduced in this thesis were demonstrated. We first introduce a microstructure with controllable parameters. This was followed by applications of convolution filtering, including an example for the transfinite interpolation and smooth distance fields. The chapter concluded by presenting concrete examples and results from the three methods introduced in chapter 3.

5.1 Main contributions

In this thesis, several contributions to distance based heterogeneous volume modelling area were made. The main contributions are listed below:

- The introduction of a general formulation for distance-based, time variant heterogeneous object modelling. This formulation is supported by several specific user-oriented operations.
- The introduction of convolution filtering for distance fields. Convolution filtering is used to create smooth distance fields, which are used in heterogeneous volume modelling to create smooth gradient material properties.
- Convolution filtering was also used to create localized smoothing of shapes. An additional volume is defined by the user to smooth the shape which intersects with that volume.
- The introduction of a method to interpolate material properties given material features within an object. Unlike transfinite interpolation, the shape conforming volumetric interpolation is shape

aware and provides more intuitive results on the base of internal distances.

- The introduction of space time transfinite interpolation, a technique to interpolate two material distributions through time. This technique is then used to demonstrate interpolation of various properties such as material and surface displacement. It is also used to create adjustable microstructures with a simple weighting parameter.
- The introduction of a new method to mix several objects together selectively based on features. An object can be made of various mixed parts of various other shapes, and blended together smoothly.

Distance fields were always considered one of the best approaches to heterogeneous object modelling. Some serious unanswered problems restricted the practical use of them. In this thesis, we identified some of these problems and proposed practical solutions and showed some applications. The majority of the solutions proposed here are implemented in a specialist purpose CAD software system by a collaborating company, Uformia AS, in Norway. However, there are still some unsolved problems and directions for improvements, some of which are shown in the next section.

5.2 Future work

A lot of research is still required to provide a sufficient framework for distance based heterogeneous volume modelling and heterogeneous volume modelling in general.

First of all, distance fields to surfaces were the primary tools employed in this thesis. However, many other fields could prove useful to describe other properties. As mentioned in chapter 3, not all properties can be defined with a function of distance and time. Also, distance to object features or elements of an object might prove useful

such as distance to the medial surface, or similar fields such as MASON (Williams and Rossignac 2005). Some problems cannot be solved with distances to boundaries, and interior distances can solve some issues when the shape of the object needs to considered. Other types of fields could be just as useful. In that regard, level sets should also be investigated for time variant, distance based properties, related to physical properties. Aside from specific fields, more tools for modelling scalar fields and vector fields are required to make heterogeneous volume modelling more accessible. *Feature*-based heterogeneous volume modelling is a useful technique, but not sufficient to provide enough control to users. Scalar field modelling can also be beneficial to geometric modelling where the scalar field quality can have a considerable impact on some operations such as blending operations and offsets. Such effects are obscure to casual users who cannot see how the field is affected by some operations.

Additionally, feature detection is largely left to the user in this thesis, however, many solutions exist to provide automatic feature detection. Typically, those techniques are designed for BRep, but volumetric techniques can be adapted from existing techniques. Automatic feature detection, for template objects, or for geometric features would greatly improve the user experience when designing heterogeneous objects.

Furthermore, many of the presented applications showed microstructure parameters using values provided by a scalar field. A useful improvement would be to iteratively improve the shape of an object given such field. This relies on automatic feature recognition, and would improve an object to fulfil a complex set of constraints. For instance, a complex object, with some constraints (a set number of support features, a fixed particular part of the shape) and some objectives such as thermal insulation, weight targets and robustness to heavy loads, could be engineered automatically provided a basic aspect of the shape.

Regarding the techniques presented in this thesis, many improvements can still be investigated:

- Smooth distance fields rely on sampling and Monte-Carlo integration. Alternative solutions should be investigated to reduce the

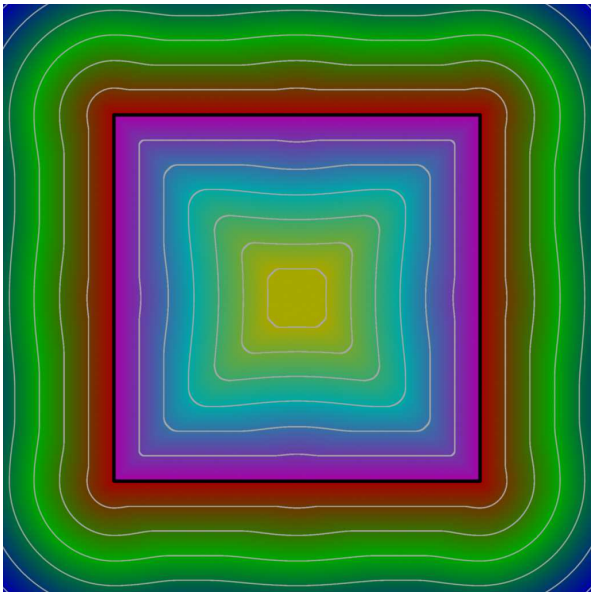


Figure 5.1: *Using C^1 continuous set-theoretic operations on the elements of a set*

number of samples, or provide better sample set to improve the final field. Additionally, a smooth field can be produced by treating each primitive in the set as an element of a union. The field is effectively C^1 continuous, but fails to provide good distance properties (see initial experiments in figure 5.1).

- In the shape conforming volumetric interpolation, the Voronoi diagram is too simplistic. An alternative solution based on ease of access, or some physically based procedure, would improve the results.

- In its current state, the space time transfinite interpolation requires the user to define the correspondence between the various parts of each object. Ideally, most features could be detected automatically, or at least suggested. Additionally, a better solution to propagate surface properties to the volume should be investigated. Meshes can use MVC, but this can be time consuming, and can only work for meshes.

- The morphological shape generation still has issues with thin features, and it is not always easy for the user to provide sensible

parameter values. Additionally, the boundary between features and feature gaps is often visible. This is because of the weighting functions changing abruptly. A smoother function would remove these undesired results. As mentioned earlier, an automatic detection of features would also improve the user experience, especially if a template is provided. Some solutions exist to build graphs from models, as shown in Zheng *et al.* (2013), which could be an initial lead.

- Interior distances are computed on a grid. There does not seem to be any solution without discretization, however, an adaptive solution could help reduce the issues currently occurring. There are two main issues to avoid; first if two parts are disconnected, but the voxel lattice cannot capture the gap (e.g., an almost closed tube loop); or two connected parts, which are not captured by the grid (e.g., two large features connected only by a very thin tube). Component analysis as presented in Fryazinov and Pasko (2015) could provide some initial ideas.

Bibliography

Abi-Ezzi S. S. and Shirman L. A., 1991. Tessellation of curved surfaces under highly varying transformations. In *Proceedings of EUROGRAPHICS*, volume 91, 385–97.

Adalsteinsson D. and Sethian J. A., 1995. A fast level set method for propagating interfaces. *Journal of computational physics*, **118**(2), 269–277.

Adzhiev V., Comninos P., Kazakov M. and Pasko A., 2005. Functionally based augmented sculpting. In *Computer Animation and Virtual Worlds*, volume 16, 25–39.

Adzhiev V., Kartasheva E., Kunii T., Pasko A. and Schmitt B., 2002. Hybrid cellular-functional modelling of heterogeneous objects. In *Journal of Computing and Information Science in Engineering, Transactions of the ASME*, volume 4, 312–322.

Adzhiev V., Kazakov M., Pasko A. and Savchenko V., 2000. Hybrid system architecture for volume modeling. In *Computers & Graphics*, volume 24, 67–78.

Adzhiev V., Cartwright R., Fausett E., Ossipov A., Pasko A. and Savchenko V., 1999. Hyperfun project: Language and Software tools for F-Rep shape modeling. volume 1, 75–100.

Autodesk , 2014. Autodesk®Maya™: 3D Animation, Visual Effects and Compositing Software. Available from: http://www.autodesk.com/maya [Accessed 10.01.2016].

Autodesk , 2015. Nastran. Available from:

http://www.autodesk.com/products/nastran/overview [Accessed 01.10.2015].

Baerentzen J. A. and Aanaes H., May 2005. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics*, **11**, 243–253.

Bar-Zeev A. 2012. Death to poly. In *RealityPrime*. http //www.realityprime.com/blog/2012/11/death-to-poly/.

Barthe L., Mora B., Dodgson N. and Sabin M., 2002. Triquadratic reconstruction for interactive modelling of potential fields. In *SMI '02: Proceedings of the Shape Modeling International 2002 (SMI'02)*, Washington, DC, USA. IEEE Computer Society, 145–153.

Bedwell E. J. and Ebert D. S., 1999. Artificial evolution of algebraic surfaces. In *Implicit Surfaces '99*, 81–88.

Beier T. and Neely S., July 1992. Feature-based image metamorphosis. *SIGGRAPH Comput. Graph.*, **26**(2), 35–42.

Belyaev A., Fayolle P.-A. and Pasko A., 2012. Signed lp-distance fields. *Computer-Aided Design*, **45**(2), 523–528.

Bhashyam S., Hoon Shin K. and Dutta D., 2000. An integrated cad system for design of heterogeneous objects. *Rapid Prototyping Journal*, **6**(2), 119–135.

Biermann H., Kristjansson D. and Zorin D., 2001. Approximate boolean operations on free-form solids. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, New York, NY, USA. ACM, 185–194.

Biswas A., Shapiro V. and Tsukanov I., 2004. Heterogeneous material modeling with distance fields. In *Comput. Aided Geom. Des.*, volume 21, Amsterdam, The Netherlands, The Netherlands. Elsevier Science Publishers B. V., 215–242.

Blinn J. F., July 1982. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, **1**, 235–256.

Bloomenthal J. and Shoemake K., July 1991. Convolution surfaces. *SIGGRAPH Comput. Graph.*, **25**, 251–256.

Bøhn J. H. and Wozny M. J., 1992. Automatic cad-model repair: Shell-closure. In *Proc. Symp. on Solid Freeform Fabrication*, 86–94.

Bojsen-Hansen M., Li H. and Wojtan C., July 2012. Tracking surfaces with evolving topology. *ACM Trans. Graph.*, **31**(4), 53:1–53:10.

Butlin G. and Stops C., 1996. CAD data repair. In *Proceedings of the 5th International Meshing Roundtable*. Citeseer, 7–12.

Caflisch R. E., 1998. Monte carlo and quasi-monte carlo methods. *Acta numerica*, **7**, 1–49.

Cameron S., Culley R. and others , 1986. Determining the minimum translational distance between two convex polyhedra. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3. IEEE, 591–596.

Campen M., Attene M. and Kobbelt L., 2012. A practical guide to polygon mesh repairing. *The Eurographics Association, pp. t4-undefined*.

Canny J. and Reif J., 1987. New lower bound techniques for robot motion planning problems. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*. IEEE, 49–60.

Carr J. C., Beatson R. K., Cherrie J. B., Mitchell T. J., Fright W. R., McCallum B. C. and Evans T. R., 2001. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, New York, NY, USA. ACM, 67–76.

Catmull E. and Clark J. *Recursively generated B-spline surfaces on arbitrary topological meshes*, 183–188. ACM, New York, NY, USA, 1998.

Cheah C., Chua C., Leong K. and Chua S., 2003a. Development of a tissue engineering scaffold structure library for rapid prototyping. part 1: investigation and classification. *The International Journal of Advanced Manufacturing Technology*, **21**(4), 291–301.

Cheah C., Chua C., Leong K. and Chua S., 2003b. Development of a tissue engineering scaffold structure library for rapid prototyping. part 2: parametric library and assembly program. *The International Journal of Advanced Manufacturing Technology*, **21**(4), 302–312.

Chen M. and Tucker J., 2000. Constructive volume geometry. In *Comput. Graph. Forum*, volume 19, 281–293.

Chen M., Jones M. W. and Townsend P., 1996. Volume distortion and morphing using disk fields. *Computers & graphics*, **20**(4), 567–575.

Chhugani J. and Kumar S., 2001. View-dependent adaptive tessellation of spline surfaces. In *Symposium on Interactive 3 D Graphics: Proceedings of the 2001 symposium on Interactive 3 D graphics*, volume 2001, 59–62.

Cho J. and Ha D., 2002. Optimal tailoring of 2d volume-fraction distributions for heat-resisting functionally graded materials using fdm. *Computer methods in applied mechanics and engineering*, **191**(29), 3195–3211.

Choi J., Sellen J. and Yap C.-K., 1997. Approximate euclidean shortest paths in 3-space. *International Journal of Computational Geometry & Applications*, **7**(04), 271–295.

Coat D., 2010. 3d coat: Voxel sculpting. Available from: http://www.3d-coat.com/tutorial/voxel-sculpting/ [Accessed 01.10.2010].

Cohen-Or D., Solomovic A. and Levin D., April 1998. Three-dimensional distance field metamorphosis. *ACM Trans. Graph.*, **17**, 116–141.

Coifman R. R., Lafon S., Lee A. B., Maggioni M., Nadler B., Warner F. and Zucker S. W., 2005. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, **102**(21), 7426–7431.

Cook R. L., Carpenter L. and Catmull E., 1987. The reyes image rendering architecture. In *ACM SIGGRAPH Computer Graphics*, volume 21. ACM, 95–102.

Copley W. Computer animation in the physical world. [Report]. Bournemouth: National Centre for Computer Animation, Bournemouth University. Unpublished, 2014.

Crane K., Weischedel C. and Wardetzky M., 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, **32**(5), 152.

Crassin C., Neyret F., Lefebvre S. and Eisemann E., 2009. Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. ACM, 15–22.

de Groot E., Wyvill B., Barthe L., Nasri A. and Lalonde P., 2014. Implicit Decals: Interactive Editing of Repetitive Patterns on Surfaces. *Computer Graphics Forum*.

DeCarlo D. and Gallier J., 1996. Topological evolution of surfaces. In *Proceedings of Graphics interface*, 194–203.

DeRose T., Kass M. and Truong T., 1998. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, New York, NY, USA. ACM, 85–94.

Dinh H. Q., Yezzi A. and Turk G., April 2005. Texture transfer during shape transformation. *ACM Trans. Graph.*, **24**(2), 289–310.

Dyllong E. and Luther W., 2000. Distance calculation between a point and a nurbs surface. Technical report, DTIC Document.

Eberly D., 2008. Distance between point and triangle in 3d. Available at `http://www.geometrictools.com` [Accessed 26 June 2011].

Ebert D. S., 2003. *Texturing & modeling: a procedural approach*. Morgan Kaufmann.

Ericson C., 2004. *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology) (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Erleben K. and Dohlmann H. 2008. 741–763. Signed distance fields using single-pass gpu scan conversion of tetrahedra. In Nguyen H., editor, *GPU Gems 3*, Addison-Wesley.

Evans A., 2006. Fast approximations for global illumination on dynamic scenes. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA. ACM, 153–171.

Farin G., 2002. *Curves and surfaces for computer aided geometric design: a practical guide*. Morgan Kaufmann Publishers Inc.

Farin G. E., 1987. *Geometric modeling: algorithms and new trends*. Number 14. Society for Industrial & Applied.

Fausett E., Pasko A. and Adzhiev V., 2000. Space-time and higher dimensional modeling for animation. In *Proceedings of the Computer Animation*. IEEE Computer Society, 156–165.

Fayolle P.-A., Pasko A., Schmitt B. and Mirenkov N., 2006. Constructive heterogeneous object modeling using signed approximate real distance functions. In *Journal of Computing and Information Science in Engineering*, volume 6, 221 – 229.

Fayolle P.-A. *Construction Of Volumetric Object Models Using Distance-Based Scalar Fields*. Thesis (PhD), University of Aizu, 2006.

Fayolle P.-A. and Pasko A., 2010. Distance to objects built with set operations in constructive solid modeling. In *Proceedings of the 13th International Conference on Humans and Computers*, HC '10, Fukushima-ken, Japan, Japan. University of Aizu Press, 41–46.

Fayolle P.-A., Pasko A. and Schmitt B. 2008. SARDF: signed approximate real distance functions in heterogeneous objects modeling, 118–141. Heterogeneous objects modelling and applications. Springer-Verlag.

Filip D., Magedson R. and Markot R., 1986. Surface algorithms using bounds on derivatives. *Computer Aided Geometric Design*, **3**(4), 295–311.

Fisher M., Fatahalian K., Boulos S., Akeley K., Mark W. R. and Hanra-

han P., 2009. Diagsplit: parallel, crack-free, adaptive tessellation for micropolygon rendering. In *ACM Transactions on Graphics (TOG)*, volume 28. ACM, 150.

Fisher S. and Lin M. C., 2001. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, New York, NY, USA. Springer-Verlag New York, Inc., 99–111.

Formnation , July 2014. Chairgenics. http://www.formnation.com/chairgenics/ [Accessed November 2015].

Freytag M., Shapiro V. and Tsukanov I., 2011. Finite element analysis in situ. *Finite Elements in Analysis and Design*, **47**(9), 957–972.

Frisken S. F., Perry R. N., Rockwood A. P. and Jones T. R., 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 249–254.

Fryazinov O., Pasko A. and Adzhiev V., March 2011. BSP-fields: An exact representation of polygonal objects by differentiable scalar fields based on binary space partitioning. In *Computer-Aided Design*, volume 43. Butterworth-Heinemann, 265–277.

Fryazinov O. and Pasko A. A., 2015. Reliable detection and separation of components for solid objects defined with scalar fields. *Computer-Aided Design*, **58**, 43–50.

Fryazinov O., Sanchez M. and Pasko A., 2015. Shape conforming volumetric interpolation with interior distances. *Computers & Graphics*, **46**(0), 149 – 155. Shape Modeling International 2014.

Fryazinov O., Vilbrandt T. and Pasko A., January 2013. Multi-scale space-variant frep cellular structures. *Comput. Aided Des.*, **45**(1), 26–34.

Fuhrmann A., Sobottka G. and Gro C., 2003. Abstract distance fields for

rapid collision detection in physically based modeling. In *Graphicon '03 Proceedings*.

Gabbrielli R., Turner I. and Bowen C. R., 2008. Development of modelling methods for materials to be used as bone substitutes. *Key Engineering Materials*, **361**, 903–906.

Gagvani N. and Silver D., May 1999. Parameter-controlled volume thinning. *CVGIP: Graph. Models Image Process.*, **61**(3), 149–164.

Galin E., Leclercq A. and Akkouche S., 2000. Morphing the BlobTree. In *Comput. Graph. Forum*, volume 19, 257–270.

Giannitelli S., Accoto D., Trombetta M. and Rainer A., 2014. Current trends in the design of scaffolds for computer-aided tissue engineering. *Acta biomaterialia*, **10**(2), 580–594.

Gilks W. R. and Wild P., 1992. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, 337–348.

Gludion , 2009. Distance to a quadratic bezier curve. Available from: http://blog.gludion.com/2009/08/distance-to-quadratic-bezier-curve.html [Accessed 09.09.2014].

Gottschalk S., Lin M. C. and Manocha D., 1996. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, New York, NY, USA. ACM, 171–180.

Green C., 2007. Improved alpha-tested magnification for vector textures and special effects. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, New York, NY, USA. ACM, 9–18.

Guéziec A., January 2001. 'meshsweeper': Dynamic point-to-polygonal-mesh distance and applications. *IEEE Transactions on Visualization and Computer Graphics*, **7**, 47–61.

Hart J. C., 1996. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, **12**(10), 527–545.

Hassouna M. S. and Farag A. A., 2007. Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **29**(9), 1563–1574.

Hastings W. K., 1970. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, **57**(1), 97–109.

Havran V. *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.

He T., Wang S. and Kaufman A., 1994. Wavelet-based volume morphing. In *Proceedings of the conference on Visualization'94*. IEEE Computer Society Press, 85–92.

Hirst A., October 2013. Light up the cigars: It is a chair. [Online; posted 9-October-2013] [Accessed September 2015].

Hongmei Z., Zhigang L. and Bingheng L., 2009. Heterogeneous object modeling based on multi-color distance field. *Materials & Design*, **30** (4), 939–946.

Hoppe H., 1996. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, New York, NY, USA. ACM, 99–108.

Hua J., He Y. and Qin H., 2004. Multiresolution heterogeneous solid modeling and visualization using trivariate simplex splines. In *Proceedings of the ninth ACM symposium on Solid modeling and applications*. Eurographics Association, 47–58.

Hughes J. F., July 1992. Scheduled fourier volume morphing. *SIGGRAPH Comput. Graph.*, **26**(2), 43–46.

ImplicitCAD , 2013. Implicitcad: Powerful, open-source, programmatic cad. Available from: http://www.implicitcad.org/ [Accessed 03.04.2014].

Jackson T., Patrikalakis N., Sachs E. and Cima M., 1998. Modeling and designing components with locally controlled composition. In *Proceed-*

ings of the Solid Freeform Fabrication Symposium. Austin, TX. USA, 259–266.

Jacobson A., Kavan L., and Sorkine-Hornung O., 2013. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, **32**(4), 33:1–33:12.

Johnson D. E. and Cohen E., 2001. Spatialized normal cone hierarchies. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, New York, NY, USA. ACM, 129–134.

Johnson D. E. and Cohen E., 2005. Distance extrema for spline models using tangent cones. In *Proceedings of Graphics Interface 2005*, GI '05, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society, 169–175.

Jones M. W., 1995. 3d distance from a point to a triangle. Technical report, Department of Computer Science, University of Wales.

Jones M. W., Baerentzen J. A. and Sramek M., July 2006. 3d distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, **12**, 581–599.

Ju T., Losasso F., Schaefer S. and Warren J., July 2002. Dual contouring of hermite data. *ACM Trans. Graph.*, **21**(3), 339–346.

Ju T., Schaefer S. and Warren J., July 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, **24**, 561–566.

Kalogerakis E., Chaudhuri S., Koller D. and Koltun V., July 2012. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, **31**(4), 55:1–55:11.

Kaufman A., 1988. Efficient algorithms for scan-converting 3d polygons. *Computers & Graphics*, **12**(2), 213–219.

Kessenich J., Baldwin D. and Rost R., 2009. The opengl shading language, version 1.40. Available from:

http://www.opengl.org/registry/doc/GLSLangSpec.Full.1.40.05.pdf [Accessed 01.09.2014].

Khoda A., Ozbolat I. T. and Koc B., 2013. Designing heterogeneous porous tissue scaffolds for additive manufacturing processes. *Computer-Aided Design*, **45**(12), 1507–1523.

Kim Y.-J., Oh Y.-T., Yoon S.-H., Kim M.-S. and Elber G., December 2011. Coons bvh for freeform geometric models. *ACM Trans. Graph.*, **30**(6), 169:1–169:8.

Knuth D. E., 1977. A generalization of dijkstra's algorithm. *Information Processing Letters*, **6**(1), 1 – 5.

Kobbelt L. P., Botsch M., Schwanecke U. and Seidel H.-P., 2001. Feature sensitive surface extraction from volume data. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, New York, NY, USA. ACM, 57–66.

Kopf J., Fu C.-W., Cohen-Or D., Deussen O., Lischinski D. and Wong T.-T., 2007. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, **26**(3), 2:1–2:9.

Kou X. Y. and Tan S. T., 2007. Heterogeneous object modeling: A review. In *Comput. Aided Des.*, volume 39, Newton, MA, USA. Butterworth-Heinemann, 284–301.

Kou X. and Tan S., 2005. A hierarchical representation for heterogeneous object modeling. *Computer-Aided Design*, **37**(3), 307 – 319. ¡ce:title¿Heterogeneous Object Models and their Applications¡/ce:title¿.

Kravtsov D. *Hybrid modelling of time-variant heterogeneous objects.* PhD thesis, Bournemouth University, 2011.

Krishnamurthy A., Khardekar R., McMains S., Haller K. and Elber G., 2008. Performing efficient nurbs modeling operations on the gpu. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, SPM '08, New York, NY, USA. ACM, 257–268.

Krishnamurthy A., McMains S. and Haller K., 2011. Gpu-accelerated

minimum distance and clearance queries. *Visualization and Computer Graphics, IEEE Transactions on*, **17**(6), 729–742.

Kumar V., Burns D., Dutta D. and Hoffmann C., 1999. A framework for object modeling. volume 31, 541 – 556.

Kumar V. and Dutta D., 1997. An approach to modeling multi-material objects. In *Proceedings of the Fourth ACM Symposium on Solid Modeling and Applications*, SMA '97, New York, NY, USA. ACM, 336–345.

Kumar V., Rajagopalan S., Cutkosky M. and Dutta D., 1998. Representation and processing of heterogeneous objects for solid freeform fabrication. In *IFIP WG5. 2 Geometric Modeling Workshop*, 7–9.

Labaco R., 2013. *Out of Hand: Materializing the Postdigital*. Black Dog Publishing.

Lal P. and Sun W., 2004. Computer modeling approach for microsphere-packed bone scaffold. *Computer-Aided Design*, **36**(5), 487 – 497.

Lamas A. and Duro R. Automatic 3d morphological design through evolution. In *IDAACS 2005. IEEE*, 564–569.

Larsen E., Gottschalk S., Lin M. C. and Manocha D., 1999. Fast proximity queries with swept sphere volumes. Technical report.

Larsen E., Gottschalk S., Lin M. C. and Manocha D., 2000. Fast distance queries with rectangular swept sphere volumes. In *Proc. of IEEE Int. Conference on Robotics and Automation*, 3719–3726.

Lazarus F. and Verroust A., 1998. Three-dimensional metamorphosis: a survey. *The Visual Computer*, **14**(8/9), 373–389.

Lee S.-H., Park T. and Kim C.-H., 2013. Primitive trees for precomputed distance queries. *Comput. Graph. Forum*, **32**(2), 419–428.

Lerios A., Garfinkle C. D. and Levoy M., 1995. Feature-based volume metamorphosis. In *SIGGRAPH '95*. ACM, 449–456.

Levi Z. and Levin D., 2014. Shape deformation via interior rbf. *Visualization and Computer Graphics, IEEE Transactions on*, **20**(7), 1062–1075.

Levy B. and Bonneel N. 2013. 349–366. Variational anisotropic surface meshing with voronoi parallel linear enumeration. In Jiao X. and Weill J.-C., editors, *Proceedings of the 21st International Meshing Roundtable*, Springer Berlin Heidelberg.

Lévy B., Petitjean S., Ray N. and Maillot J., July 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, **21**(3), 362–371.

Lin M. and Gottschalk S., 1998. Collision detection between geometric models: A survey. In *Proc. of IMA conference on mathematics of surfaces*, volume 1, 602–608.

Lin M. C. and Canny J. F., 1991. Efficient algorithms for incremental distance computation. In *IEEE conference on robotics and automation*, volume 10081014.

Lipman Y., Levin D. and Cohen-Or D., August 2008. Green coordinates. *ACM Trans. Graph.*, **27**(3), 78:1–78:10.

Liu H., Maekawa T., Patrikalakis N., Sachs E. and Cho W., 2004. Methods for feature-based design of heterogeneous solids. *Computer-Aided Design*, **36**(12), 1141–1159.

Liu H. *Algorithms for design and interrogation of functionally graded material solids*. PhD thesis, Massachusetts Institute of Technology, 2000.

Loop C. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, The University of Utah, August 1987.

Loop C. and Blinn J., 2005. Resolution independent curve rendering using programmable graphics hardware. *ACM Transactions on Graphics (TOG)*, **24**(3), 1000–1009.

Lorensen W. E. and Cline H. E., July 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, volume 21. ACM Press, 163–169.

Lu L., Sharf A., Zhao H., Wei Y., Fan Q., Chen X., Savoye Y., Tu C.,

Cohen-Or D. and Chen B., 2014. Build-to-last: Strength to weight 3d printed objects. *ACM Transactions on Graphics (TOG)*, **33**(4), 97.

Ma Y. L. and Hewitt W., 2003. Point inversion and projection for NURBS curve and surface: Control polygon approach. *Computer Aided Geometric Design*, **20**(2), 79 – 99.

Martin W. and Cohen E., 2001. Representation and extraction of volumetric attributes using trivariate splines: a mathematical framework. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*. ACM, 234–240.

McLoughlin L., Fryazinov O., Moseley M., Sanchez M., Adzhiev V., Comninos P. and Pasko A., 2016. Virtual sculpting and 3d printing for young people with disabilities. *IEEE Computer Graphics and Applications*.

Melchels F. P., Bertoldi K., Gabbrielli R., Velders A. H., Feijen J. and Grijpma D. W., 2010. Mathematically defined tissue engineering scaffold architectures prepared by stereolithography. *Biomaterials*, **31** (27), 6909–6916.

Michalatos P. and Payne A., 2015. Monolith 3d. Available from: http://www.monolith.zone/download/ [Accessed 15.09.2015].

Moakher M., January 2002. Means and averaging in the group of rotations. *SIAM J. Matrix Anal. Appl.*, **24**(1), 1–16.

Morokoff W. J. and Caflisch R. E., 1995. Quasi-monte carlo integration. *Journal of computational physics*, **122**(2), 218–230.

Morse B., Yoo T., Rheingans P., Chen D. and Subramanian K., May 2001. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *Shape Modeling and Applications, SMI 2001 International Conference on.*, 89–98.

Nguyen H., 2007. *GPU GEMS 3*. Addison-Wesley Professional.

Nielson G. M., January 1993. Scattered data modeling. *IEEE Comput. Graph. Appl.*, **13**(1), 60–70.

Nielson G. M., 2004. Dual marching cubes. In *Proceedings of the conference on Visualization '04*, VIS '04, Washington, DC, USA. IEEE Computer Society, 489–496.

Ohtake Y., Belyaev A., Alexa M., Turk G. and Seidel H.-P., 2003. Multi-level partition of unity implicits. In *ACM Transactions on Graphics (Proc. SIGGRAPH'03)*, volume 22. ACM, 463–470.

Papadimitriou C. H., 1985. An algorithm for shortest-path motion in three dimensions. *Information Processing Letters*, **20**(5), 259–263.

Park S.-M., Crawford R. H. and Beaman J. J., 2001. Volumetric multi-texturing for functionally gradient material representation. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*. ACM, 216–224.

Pasko A., Adzhiev V., Sourin A. and Savchenko V., 1995. Function representation in geometric modeling: Concepts, implementation and applications. In *The Visual Computer*, number 11, 429–446.

Pasko A., Savchenko A. and Savchenko V., 1996. Polygon-to-function conversion for sweeping. In *Proceedings of Implicit Surfaces*, volume 96, 163–171.

Pasko A. and Adzhiev V. 2004. 132–160. Function-based shape modeling: Mathematical framework and specialized language. In Winkler F., editor, *Automated Deduction in Geometry*, volume 2930 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg.

Pasko A., Adzhiev V., Schmitt B. and Schlick C., 2001. Constructive hypervolume modeling. *Graphical Models*, **63**(6), 413 – 442.

Pasko A., Fryazinov O., Vilbrandt T., Fayolle P.-A. and Adzhiev V., 2011a. Procedural function-based modelling of volumetric microstructures. *Graphical Models*, **73**(5), 165 – 181.

Pasko G., Pasko A. and Kunii T., 2004. Space-time blending. In *Computer Animation and Virtual Worlds*, volume 15, 109–121.

Pasko G., Pasko A., Vilbrandt T., Lixandrão Filho A. L. and da Silva

J. V. L., 2011b. Ascending in space dimensions: Digital crafting of MC Escher's graphic art. *Leonardo*, **44**(5), 411–416.

Pasko G., Pasko A. and Kunii T., 2005. Bounded blending for function-based shape modeling. *Computer Graphics and Applications, IEEE*, **25**(2), 36–45.

Payne B. A. and Toga A. W., January 1992. Distance field manipulation of surface models. *IEEE Comput. Graph. Appl.*, **12**, 65–71.

Pedersen H. K., 1995. Decorating implicit surfaces. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, New York, NY, USA. ACM, 291–300.

Pekerman D., Elber G. and Kim M.-S., 2008. Self-intersection detection and elimination in freeform curves and surfaces. *Computer-Aided Design*, **40**(2), 150 – 159.

Perlin K., July 1985. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, volume 19. ACM Press, 287–296.

Peytavie A., Galin E., Grosjean J. and Merillou S., 2009. Procedural generation of rock piles using aperiodic tiling. In *Computer Graphics Forum*, volume 28. Wiley Online Library, 1801–1809.

Pharr M. and Humphreys G., 2004. *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

Piegl L., 1991. On nurbs: a survey. *IEEE Computer Graphics and Applications*, **11**(1), 55–71.

Piegl L. and Tiller W., 1997. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA.

Piegl L. A. and Tiller W., 1999. Computing offsets of nurbs curves and surfaces. *Computer-Aided Design*, **31**(2), 147 – 156.

Qian X. and Dutta D., 2003a. Design of heterogeneous turbine blade. *Computer-Aided Design*, **35**(3), 319–329.

Qian X. and Dutta D., 2003b. Physics-based modeling for heterogeneous objects. *Journal of mechanical Design*, **125**(3), 416–427.

Quilez I., 2008. Rendering worlds with two triangles. Available at `http://www.iquilezles.org/www/material/nvscene2008` [Accessed 15 December 2011].

Quinlan S., 1994. Efficient distance computation between non-convex objects. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 3324–3329.

Rajab K., Piegl L. A. and Smarodzinava V., 2013. CAD model repair using knowledge-guided NURBS. *Engineering with Computers*, **29**(4), 477–486.

Rajab K., Piegl L. and Smarodzinava V., 2012. CAD model repair using knowledge-guided NURBS. *Engineering with Computers*, 1–10.

Requicha A., 1996. *Geometric Modeling: A First Course*.

Requicha A. G., December 1980. Representations for rigid solids: Theory, methods, and systems. *ACM Comput. Surv.*, **12**(4), 437–464.

Ricci A., 1973. A Constructive Geometry for Computer Graphics. In *The Computer Journal*, volume 16, 157–160.

Rosenfeld A. and Pfaltz J. L., October 1966. Sequential operations in digital picture processing. *J. ACM*, **13**, 471–494.

Rossignac J., Fudos I. and Vasilakis A., February 2013. Direct rendering of boolean combinations of self-trimmed surfaces. *Comput. Aided Des.*, **45**(2), 288–300.

Rossignac J. and Requicha A., 1984. Constant-radius blending in solid modelling.

Rustamov R., Lipman Y. and Funkhouser T., July 2009. Interior distance using barycentric coordinates. *Computer Graphics Forum (Symposium on Geometry Processing)*, **28**(5).

Rvachev V., 1982. Theory of r-functions and some applications [in russian]. *Naukova Dumka, Kiev*.

Rvachev V. L., Sheiko T. I., Shapiro V. and Tsukanov I., 2001. Transfinite interpolation over implicitly defined sets.

Samanta K. and Koc B., 2005. Feature-based design and material blending for free-form heterogeneous object modeling. *Computer-Aided Design*, **37**(3), 287 – 305. Heterogeneous Object Models and their Applications.

Sanchez M., Fryazinov O., Adzhiev V., Comninos P. and Pasko A., 2014. Space-time transfinite interpolation of volumetric material properties. *Visualization and Computer Graphics, IEEE Transactions on*, **PP** (99), 1–1.

Sanchez M., Fryazinov O., Fayolle P.-A. and Pasko A., 2015. Convolution filtering of continuous signed distance fields for polygonal meshes. *Computer Graphics Forum*, **34**(6), 277–288.

Sanchez M., Fryazinov O. and Pasko A., 2012. Efficient evaluation of continuous signed distance to a polygonal mesh. In *Proceedings of the 28th Spring Conference on Computer Graphics*, SCCG '12, New York, NY, USA. ACM, 101–108.

Sanchez M., Fryazinov O., Vilbrandt T. and Pasko A., 2013. Morphological shape generation through user-controlled group metamorphosis. *Computers & Graphics*, **37**(6), 620 – 627. Shape Modeling International (SMI) Conference 2013.

Savchenko V., Pasko A., Okunev O. and Kunii T., 1995. Function representation of solids reconstructed from scattered surface points and contours. In *Comput. Graph. Forum*, volume 14, 181–188.

Schmidt R., Wyvill B., Sousa M. C. and Jorge J. A., 2006. Shapeshop: sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA. ACM.

Schmidt R. and Wyvill B., 2005. Implicit sweep surfaces. *Department of Computer Science. University of Calgary.*

Schmitt B., Pasko A., Adzhiev V. and Schlick C., December 2001. Con-

structive texturing based on hypervolume modeling. In *The booktitle of Visualization and Computer Animation*, volume 12, 297–310.

Schroeder W. J., Lorensen W. E. and Linthicum S., 1994. Implicit modeling of swept surfaces and volumes. In *VIS '94: Proceedings of the conference on Visualization '94*, Los Alamitos, CA, USA. IEEE Computer Society Press, 40–45.

Schumacher C., Bickel B., Rys J., Marschner S., Daraio C. and Gross M., 2015. Microstructures to control elasticity in 3d printing. *ACM Transactions on Graphics (TOG)*, **34**(4), 136.

Sederberg T. W., Zheng J., Bakenov A. and Nasri A., July 2003. T-splines and t-nurccs. *ACM Trans. Graph.*, **22**(3), 477–484.

Shapeways , 2015. Svx format. Available from: http://abfab3d.com/svx-format/ [Accessed 01.11.2015].

Shapiro V., 1991. Theory of r-functions and applications: A primer. Technical report, Cornell University.

Shapiro V., 2007. Semi-analytic geometry with R-functions. *Acta Numerica*, **16**, 239–303.

Shewchuk J. R., 1998. Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry*. ACM, 86–95.

Shin K.-H. and Dutta D., 2001. Constructive representation of heterogeneous objects. *Journal of Computing and Information Science in Engineering*, **1**(3), 205–217.

Shoemake K. and Duff T., 1992. Matrix animation and polar decomposition. In *Proceedings of Graphics interface*, 258–264.

Si H. and TetGen A., 2006. A quality tetrahedral mesh generator and three-dimensional delaunay triangulator. *Weierstrass Institute for Applied Analysis and Stochastic, Berlin, Germany*.

Sigg C., Peikert R. and Gross M., 2003. Signed distance transform using graphics hardware. In *Proceedings of the 14th IEEE Visualization 2003*

(VIS'03), VIS '03, Washington, DC, USA. IEEE Computer Society, 12–.

Silva S., Fayolle P.-A., Vincent J., Pauron G., Rosenberger C. and Toinard C., 2005. Evolutionary computation approaches for shape modelling and fitting. In *Proceedings of the 12th Portuguese conference on Progress in Artificial Intelligence*, EPIA'05. Springer-Verlag, 144–155.

Sims K., January 1994. Evolving 3d morphology and behavior by competition. *Artif. Life*, **1**(4), 353–372.

Siu Y. K. and Tan S. T., 2002a. Modeling the material grading and structures of heterogeneous objects for layered manufacturing. *Computer-Aided Design*, **34**(10), 705–716.

Siu Y. and Tan S., 2002b. Source-based heterogeneous solid modeling. *Computer-Aided Design*, **34**(1), 41 – 55.

Smets-Solanes J.-P., 1996. Vector field based texture mapping of animated implicit objects. *Computer Graphics Forum*, **15**(3), 289–300.

Stinson L., October 2013. Designer uses algorithm to genetically breed chairs. [Online; posted 24-October-2013] [Accessed September 2015].

Strain J., July 1999. Fast tree-based redistancing for level set computations. *J. Comput. Phys.*, **152**(2), 664–686.

Takahashi S., Kokojima Y. and Ohbuchi R., 2001. Explicit control of topological transitions in morphing shapes of 3d meshes. In *PG'01*. IEEE Computer Society, 70–.

Tigges M. and Wyvill B., 1998. Texture mapping the blobtree. In *In Proceedings of the Third International Workshop on Implicit Surfaces*, 123–130.

Tsitsiklis J. N., 1995. Efficient algorithms for globally optimal trajectories. *Automatic Control, IEEE Transactions on*, **40**(9), 1528–1538.

Turk G. and O'Brien J. F., 1999a. Shape transformation using variational

implicit functions. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 335–342.

Turk G. and O'Brien J. F., 1999b. Variational implicit surfaces.

Voyatzis C., October 2013. The future of furniture: Breeding chairs. [Online; posted 09-October-2013] [Accessed September 2015].

Wang L., Yu Y., Zhou K. and Guo B., December 2011. Multiscale vector volumes. *ACM Trans. Graph.*, **30**(6), 167:1–167:8.

Wang M. Y. and Wang X., 2005. A level-set based variational method for design and optimization of heterogeneous objects. *Computer-Aided Design*, **37**(3), 321–337.

Wejrzanowski T., Skibinski J., Szumbarski J. and Kurzydlowski K., 2013. Structure of foams modeled by laguerrevoronoi tessellations. *Computational Materials Science*, **67**(0), 216 – 221.

Weller R. and Zachmann G., 2010. Protosphere: A gpu-assisted prototype guided sphere packing algorithm for arbitrary objects. In *ACM SIGGRAPH ASIA 2010 Sketches*. ACM, 8.

Williams J. and Rossignac J., July 2005. Mason: morphological simplification. *Graph. Models*, **67**(4), 285–303.

Worley S., 1996. A cellular texture basis function. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, New York, NY, USA. ACM, 291–294.

Wu X., Liu W. and Wang M. Y., 2005. Feature based modeling of heterogeneous objects. *Multidiscipline Modeling in Materials and Structures*, **1**(4), 341–366.

Wyvill B., 1993. Metamorphosis of implicit surfaces. In *Modeling, Visualizing and Animating with Implicit Surfaces*.

Wyvill B., Guy A. and Galin E., 1999. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. In *Computer Graphics Forum*, number 2, 149–158.

Wyvill B., McPheeters C. and Wyvill G., 1986. Animating Soft Objects. In *The Visual Computer*, volume 2, 235–242.

Xu H., Li Y., Chen Y. and Barbivč J., 2015. Interactive material design using model reduction. *ACM Transactions on Graphics (TOG)*, **34** (2), 18.

Xu K., Zhang H., Cohen-Or D. and Chen B., July 2012. Fit and diverse: set evolution for inspiring 3d shape galleries. *ACM Trans. Graph.*, **31** (4), 57:1–57:10.

Yang H. and Jüttler B., 2007. 3d shape metamorphosis based on t-spline level sets. *The Visual Computer*, **23**(12), 1015–1025.

Yen L., Fouss F., Decaestecker C., Francq P. and Saerens M. 2007. 1037–1045. Graph nodes clustering based on the commute-time kernel. In *Advances in Knowledge Discovery and Data Mining*, Springer.

Yngve G. and Turk G., October 2002. Robust creation of implicit surfaces from polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics*, **8**, 346–359.

Yoshizawa S., Belyaev A. and Seidel H.-P., 2004. A fast and simple stretch-minimizing mesh parameterization. In *Proceedings of the Shape Modeling International 2004*, SMI '04, Washington, DC, USA. IEEE Computer Society, 200–208.

Zhang X.-J., Chen K.-Z. and Feng X.-A., 2004. Optimization of material properties needed for material design of components made of multi-heterogeneous materials. *Materials & design*, **25**(5), 369–378.

Zheng Y., Cohen-Or D. and Mitra N. J., 2013. Smart variations: Functional substructures for part compatibility. *Computer Graphics Forum (Eurographics)*, **32**(2pt2), 195–204.

Zhou M., Xi J. and Yan J., 2004. Modeling and processing of functionally graded materials for rapid prototyping. *Journal of Materials Processing Technology*, **146**(3), 396–402.

Appendix A

Calculating interior distances

The algorithm 1 is a modification of Djikstra's algorithm which corrects the error accumulation for interior distances by keeping track of additional information. Each voxel A in the grid stores its minimal distance to the seed and the coordinates of the closest voxel to the seed still visible by A. The closest voxel to the seed still visible is called the *root* voxel. This way, the shortest path from a voxel to the seed is simply the path from the voxel to its root and the shortest path of the root to the seed. Therefore, the interior distance is simply the sum of the distance between a voxel and its root voxel with the interior distance of the root voxel.

The function $is_Visible$ in the algorithm can be implemented by using the DDA algorithm on a voxel grid to see if the voxel being processed has visibility of another voxel.

Algorithm 1: Interior propagation

Input: *seeds*: Initial seeds
G: A grid
S: Shape which defines the boundaries
Output: Seeds for exterior propagation O

```
1  F: Priority queue of cell coordinates ;
2  foreach p in seeds do
3  |  insert p to F;
4  end
5  while F not empty do
6  |  cell coordinates current retrieved from front;
7  |  currentvalue ← value at current in G;
8  |  currentroot ← value at current in G;
9  |  rootvalue ← value at currentroot in G;
10 |  foreach Neighbour coordinate neighbour do
11 |  |  if neighbour outside S then
12 |  |  |  frontdistance ← distance(neighbour, current);
13 |  |  |  recordeddistance ← value at neighbour in G;
14 |  |  |  if frontdistance ≤ recordeddistance then
15 |  |  |  |  value at neighbour in G ← frontdistance;
16 |  |  |  end
17 |  |  |  insert unique neighbour in O;
18 |  |  else
19 |  |  |  if isVisible(neighbour,currentroot) then
20 |  |  |  |  frontdistance ←
   |  |  |  |  distance(neighbour, currentroot) + rootvalue;
21 |  |  |  |  neighbourroot ← currentroot;
22 |  |  |  else
23 |  |  |  |  frontdistance ←
   |  |  |  |  distance(neighbour, current) + currentvalue;
24 |  |  |  |  neighbourroot ← current;
25 |  |  |  end
26 |  |  |  recordeddistance ← value at neighbour in G;
27 |  |  |  if recordeddistance ≤ frontdistance then
28 |  |  |  |  continue;
29 |  |  |  end
30 |  |  |  value at neighbour in G ← frontdistance;
31 |  |  |  root at neighbour in G ← neighbourroot;
32 |  |  |  insert neighbour to F;
33 |  |  end
34 |  end
35 end
```

Appendix B

Disconnected components and Voronoi diagrams

Disconnected components can appear when a Voronoi diagram using Euclidean distances is intersected with an object (see figure B.1a). If an object needs to be partitioned, a Voronoi diagram can be used, but Euclidean distances will only work with convex shapes. For concave shapes, interior distances need to be used if we want cells to be made of a single component (such as figure B.1b).

We want to prove that a Voronoi diagram using interior distances within an object creates cells which are made of a single component, given that the material features (Voronoi seeds, which can be extended to volumes in our case) are entirely inside the shape, and made of a single component.

Suppose a Voronoi diagram using interior distances within an object creates cells with disconnected parts.

For the shortest path q which connects a point A to its closest seed v_i, any point of q shares the same closest seed, otherwise, q could be shortened.

By definition, a cell c_i is defined by the set of points for which the closest seed is the seed v_i. Let p be a point from one of the components of a cell c_i which is not connected to the component which contains the

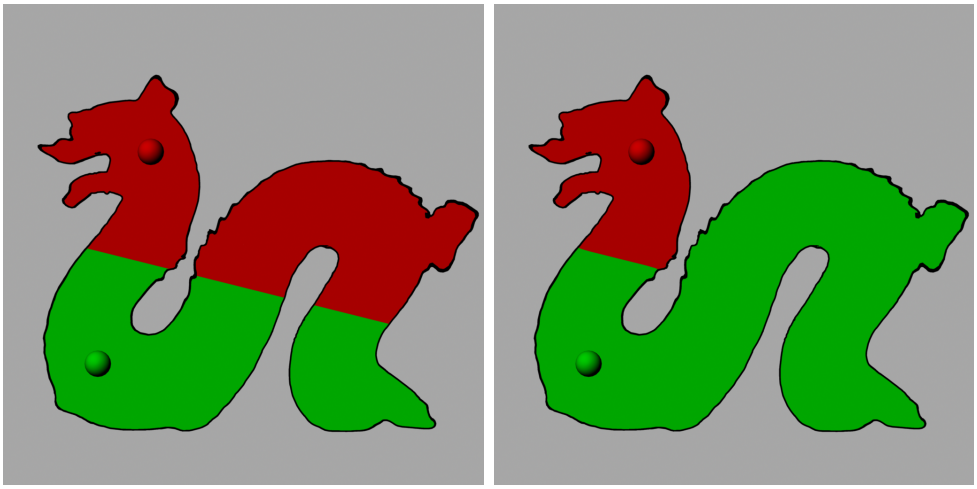


Figure B.1: *A cross section of the Stanford Dragon with two Voronoi seeds. (a) using Euclidean distances, (b) using interior distances. A Voronoi diagram of an object using Euclidean distances creates disconnected components.*

seed v_i, and let ω the path which connects p and v_i. For any point on ω, its closest seed is v_i. However, since the components are disconnected, the path to the closest seed needs to cross the boundaries of the cell c_i, and therefore, some of those points have a different closest seed (c_j) which is contradictory.

Therefore, the path ω can only be entirely in the same cell and therefore a cell can only be made of a single component.

Appendix C

Generation of foam microstructures

Subsection 2.4.3 showed that procedural microstructures can be useful in the creation of lightweight and robust objects. However, adaptive polymer foams are more popular in material engineering, and are not easily produced by a procedural function. Typical microstructures can be adaptive and parameters can be functions of space to better fit a shape and its constraints (see sections 4.4 and 4.3 or Pasko *et al.* (2011a) and Fryazinov *et al.* (2013)). Foams are more robust than a regular lattice and can be used to improve several properties of an object such as heat conservation, weight, or insulation or simply robustness (Lu *et al.* 2014).

The objective here is to reproduce different foam types as internal structures. The objective is to either create a geometrical structure which fits into a shape, or a function defining various materials, gradual or composite.

A Voronoi wall structure removes Voronoi-shaped cells leaving only the walls in-between the cells with some thickness. The thicker the wall, the more robust a cell would be, but this also requires more material. This means that the user needs to be able to control this parameter across the object to remove as much material as possible in volumetric regions with minimal stress. Similarly, the edges between the walls can

be blended together to increase the robustness of the structure. Once again, this parameter can be controlled so that the blending material is only added where it is needed.

The definition of a Voronoi diagram and its function representation was already detailed in section 3.3. The same definition can be used with only minor changes to fit the requirements defined above. The function in equation (3.18) defined a single cell of a Voronoi diagram, where any point inside the cell was considered solid.

The equation is modified to create solid walls between the cells. The modifications are:

- Offsets need to be applied to the cells to create a solid around the boundaries between the cells.
- The function needs to consider all the cells to create one solid.
- The sign need to be flipped so that the cells are hollow.

Therefore, the equation for the Voronoi structure is:

$$m(\mathbf{p}) = -[(v_c(\mathbf{p}) - t_i) \bigvee_{i=1\ldots n_q} (v_{q_i}(\mathbf{p}) - t_{q_i})] \tag{C.1}$$

where:

- v_i is the function which represent an individual cell of the Voronoi diagram,
- c is the index of the cell the point $\mathbf{p}$ belongs to,
- n_q is the number of neighbours and
- q_i is the i-th neighbour of the cell c,
- t_i is the partial wall thickness for a cell.

The details of the wall thickness t_i are in the following section C.1. The cells v_i can be rounded to improve the robustness of the structure, and the details are provided in section C.2. The section C.3 shows how the parameters of the microstructures can be controlled easily. Finally, the seeds of the Voronoi diagram is covered in section C.4.

C.1 Wall thickness

Similarly to the work presented in section 3.3, the cells are offset inwards by subtracting a value t_i. The thickness of the wall is hardly meaningful if it changes across a cell. In fact, a high frequency function could cause abnormal behaviour on the walls and weaken the overall structure. Instead, the total thickness of the wall is set by two values assigned to the cells on either sides of the wall. The offset values of neighbouring cells can be different, but the wall will be as thick as the sum of the neighbour cells thickness values. So, for instance, if two neighbouring cells A and B, have a wall thickness of 0.1 and 0.2 respectively, then the wall between A and B will be 0.3.

The choice of the union R-function impacts the distance properties of the overall function, therefore using the $\underset{2}{\overset{0}{R}}$ class of R-functions (Rvachev 1982) is favoured. If rounded unions or rounded intersections are required, the formulation from ImplicitCAD (2013) is favoured since it provides fixed radius blending, and yet maintains good distance properties (see section 2.4.1).

C.2 Roundness

In equation 3.18, intersections were used to create the Voronoi cell. In order to control the roundness of a cell, blended intersections are employed where the roundness can be controlled and a value exists so that the blended intersection is equivalent to the ordinary intersection. The rounded union presented in ImplicitCAD (2013) fulfils those requirements. This operator is C^1 continuous, has reasonable distance properties and is controlled by a simple roundness parameter. The equation

for the rounded intersection is reproduced here for convenience:

$$f_1 \wedge_{rounded} f_2 = \begin{cases} \min(f_1, f_2) & \text{if } |f_1 - f_2| \geq r \\ f_2 + r \cdot \sin\left(\frac{\Pi}{4} + asin(\frac{f_1 - f_2}{r\sqrt{2}})\right) - 2 & \text{if } |f_1 - f_2| < r \end{cases} \tag{C.2}$$

Similarly to the thickness parameter, the roundness parameter of a cell is defined per cell in order to prevent unexpected results due to a high frequency function.

C.3 Parameter ranges

The roundness parameter of the rounded union depends on the scale of the object and cells which means that it can be unintuitive for the designer to define, especially for a large number of cells. This can be particularly problematic if the cells do not have uniform sizes. Instead, the normalization of the parameters (of roundness and wall thickness) is proposed that adapts in respect to the size of the cell. This way both parameters have a known range $[0, 1]$. For the wall thickness, 0 translates to non-existent wall (no thickness), which only has meaning for heterogeneous volumetric modelling, and 1 means maximum thickness where the user would expect it to be the smallest value of the wall thickness which completely fills the cells. For the roundness parameter, 0 would mean no blending, and 1 would mean as round as possible where the cell is almost spherical for a uniform distribution.

To remap the parameters per cell, the maximum meaningful value needs to be identified. The radius of the inscribed circle is a great heuristic. Finding the radius of the inscribed circle of a Voronoi cell can be achieved using the gradient descent technique on a distance field of the Voronoi cell function. There are no risks of local maximum trapping (assuming Euclidean distances are used), since there is only one maximum unless the shape has parallel planes, in which case they all share the same maximum value.

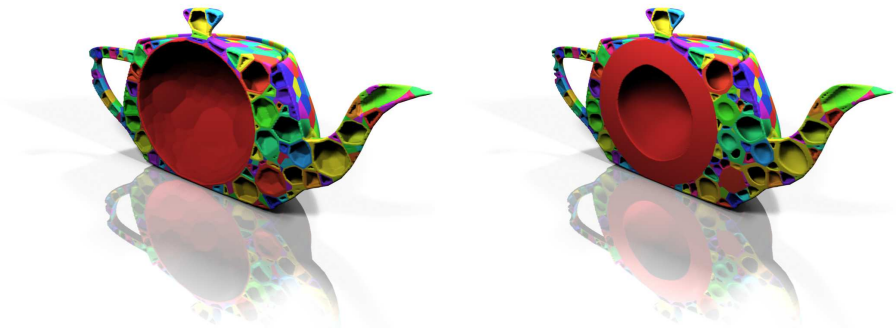
(a) (b)

Figure C.1: *Sphere packing on the teapot (a) and the Buddha (b)*

C.4 Seeds distribution

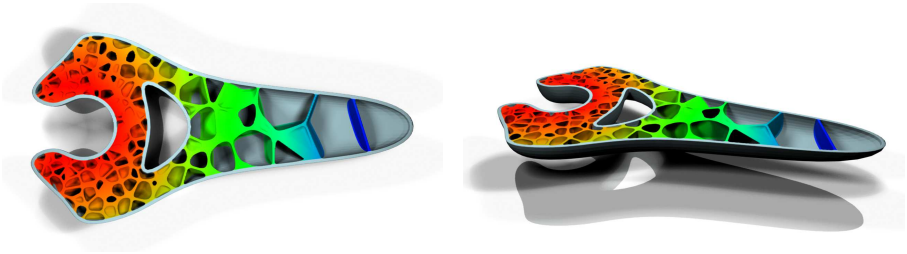
A popular solution for polymer foam mentioned in Lal and Sun (2004) is sphere packing combined with Laguerre-Voronoi tessellation, which allows for weighted seeds, where higher weights correspond to larger cells. The weight can be seen as the radius of a sphere. Each seed of the Voronoi diagram becomes a sphere. This requires minimal modifications to the algorithms presented above, since the cell boundaries remain planar and are only shifted to account for the weight imbalance. In order to perform sphere packing on arbitrary shapes, the method detailed in Weller and Zachmann (2010) is used. Here sphere packing means fitting the largest spheres first and fitting more spheres by decreasing the radius until the shape is sufficiently filled. An upper bound on the radius of the sphere can be used to limit the maximum size of the cells. The results of sphere packing can be seen in figure C.1. The polymer foam resulting from sphere packing can be seen in figure C.2.

More importantly, a designer might want to control the foam density in the function of the region of the shape. Using the Laguerre-Voronoi tessellation, the user provides a cell size function. This function is related to heterogeneous volumetric modelling and gradient material properties and therefore, all the methods introduced in this document are relevant. To provide the seeds and their weights, rejection sampling is used with



(a) (b)

Figure C.2: *Foam using set theoretic operations (a) and blending operations (b)*



(a) (b)

Figure C.3: *A simple key shape driven by a radius function. The radius is small along a stress path, and larger as it goes further away from it within the shape using interior distances.*

Poisson disc sampling. In this case, each sample stores its radius and so when a new sample point is tested, a collision check is performed to make sure the new sample does not collide with any of the previous samples. The sampling stops when the tests fail repetitively many times. Voronoi Relaxation is applied afterwards to improve on the sampling distribution. One of the caveats of this method is that the sample set can fail to accurately represent a high frequency function and a large radius tends to cover small radius regions. Such function cannot be accurately sampled because a single sample could overlap several extrema of the function. A solution to prevent the user from providing such function is needed, or at least issue warnings when the function is not accurately represented, however, it is out of the scope of this work.

In figure C.3, a stress path was created by finding the shortest path between two points in the shape. Next, the interior distance field of this path was created and is visualized by the colouring. Red, green and blue colours represent close, medium and far values respectively. A transfer function is then used to change the interior distance into a radius. The microstructure is then generated. Larger walls and more blending are used in the vicinity of the stress path.

Appendix D

Main algorithms for distance based heterogeneous volumetric modelling

In this appendix, we outline algorithms to implement the various methods and techniques described in chapter 3 for distance based heterogeneous volumetric modelling. Some are trivial and do not require much explanation, but are shown for convenience.

D.1 Space time transfinite interpolation

Space time transfinite interpolation as presented in section 3.4 is straight forward to implement and is based only on the two signed distances of the objects. Note that the algorithm only provides two weights, but does not perform the interpolation. This also means that the attributes can be functions of space, but the algorithm will not change.

Algorithm 2 shows the simplest case of space time transfinite interpolation. For partitioned objects, similar process has to be used, but the transformation algorithm also needs to be available.

Algorithm 2: Space time transfinite interpolation

Input:
f_1: Signed distance to source object G_1
f_2: Signed distance to target object G_2
t: Time value for interpolation in range $[0, 1]$
α: time gap
β: object balance in range $[0, 1]$
Output: Property weights w_1 and w_2

$m_1 \leftarrow f_1 \cdot 2(1-\beta)$;
$m_2 \leftarrow f_2 \cdot 2\beta$;
$g_1 \leftarrow m_1 \wedge -\alpha t$;
$g_2 \leftarrow m_2 \wedge (\alpha - 1)t$;
$gsum \leftarrow g_1 + g_2$;
$w_1 \leftarrow \frac{g_2}{gsum}$;
$w_2 \leftarrow \frac{g_1}{gsum}$;

D.2 Shape conforming volumetric interpolation

Shape conforming volumetric interpolation relies on the inverse distance weighting formula for n sources. The algorithm 3 describes how to compute the weights for each source given the set of n distances.

Algorithm 4 builds on interior distances to first construct Voronoi cells with offsets (v_i) and then applies the interpolation. If the point is inside the cell feature, the value of v_i is set to zero. Given that the offsets are not set to zero, the cell features cannot overlap. Therefore, we cannot have more than one value v_i equal to zero. When v_i is equal to zero, then this means that it is within the cell feature, and therefore, no interpolation is required.

D.3 Feature-based group metamorphosis

The method for feature-based group metamorphosis described in section 3.5 allows to combine and mix various parts of various objects.

The algorithm 5 shows how the group metamorphosis can be imple-

Algorithm 3: Transfinite interpolation for n sources

Input:
n: Number of material features
d: Set of n distances to each feature
A: Set of n properties, where A_i is the property of the i-th object
Output: Set of weights w and attribute A_o

$distsum \leftarrow 0$;
$distprod \leftarrow 1$;
foreach *Distance* d_i **do**
    $distsum \leftarrow distsum + d_i$;
    $distprod \leftarrow distprod \cdot d_i$;
end
$nums$ array of n elements;
$denominator \leftarrow 0$;
foreach *Distance* d_i **do**
    $nums_i \leftarrow \frac{prod}{d_i}$;
    $denominator \leftarrow denominator + T_i$;
end
foreach *Distance* d_i **do**
    $w_i \leftarrow \frac{nums_i}{denominator}$;
    $A_o \leftarrow A_i \cdot w_i$;
end

Algorithm 4: Shape conforming interpolation algorithm

Input:
n: Number of material features
d: Set of n interior distances to each feature
A: Set of n properties, where A_i is the property of the i-th object
s: Transition width matrix, where $s_{i,j} = s_{j,i}$
Output: Set of weights w and attribute A_o

```
1  v, an array of n elements;
2  for i = 1, i <= n, i ← i + 1 do
3  |   v_i ←undefined;
4  |   for j = 1, j <= n, j ← j + 1 do
5  |   |   if i ≠ j then
6  |   |   |   h_{i,j} ← d_i − d_j;
7  |   |   |   b_{i,j} ← h_{i,j} + s_{i,j};
8  |   |   |   if cell is undefined then
9  |   |   |   |   v_i ← b_{i,j};
10 |   |   |   else
11 |   |   |   |   v_i ← v_i ∧ b_{i,j};
12 |   |   |   end
13 |   |   end
14 |   end
15 |   v_i ← max(0, v_i);
16 end
17 transfiniteinterpolation(n, v, A)
```

Algorithm 5: Group metamorphosis with affine transformations

Input:
P: Position
k: Number of objects
f_j: jth object's field
$E_{i,j}$: Original feature i of object j
Output: Field value at P

$ret \leftarrow 0$;
$denom \leftarrow 0$;
foreach *Feature* i **do**
    $E'_i \leftarrow$ Interpolated feature i ;
    $R \leftarrow$ relative position of P in regards to feature E'_i ;
    $accum_{field} \leftarrow 0$;
    foreach *Object* j **do**
        $Q \leftarrow$ absolute position of R in regards to $E_{i,j}$;
        $accum_v \leftarrow f_j(Q) \cdot v_i$;
        $accum_{field} \leftarrow accum_{field} + accum_v$;
    end
    $d_i \leftarrow$ distance from P to feature E_i ;
    $w_i \leftarrow w(d_i)$;
    $ret \leftarrow ret + (accum_{field} + o_i) \cdot w_i$;
    $denom \leftarrow denom + w_i$;
end
return $\frac{ret}{denom}$;

mented when affine transformations are applied to feature elements.

Appendix E

List of publications

Sanchez, M., Fryazinov, O. and Pasko, A., 2012. Efficient Evaluation of Continuous Signed Distance to a Polygonal Mesh. In *SCCG '12: Proceedings of the 28th Spring conference on Computer graphics*, Comenius University, Bratislava 2012, Pages 111-118.

Sanchez, M., Fryazinov, O., Vilbrandt, T. and Pasko, A., 2013. Morphological shape generation through user-controlled group metamorphosis, *Computer & Graphics*, Volume 37, Issue 6, October 2013, Pages 620-627.

Fryazinov, O., Sanchez, M., Adzhiev, V. and Pasko, A., 2013. Time-variant Volumetric Colors for Metamorphosis, *Eurographics 2013-Posters*, The Eurographics Association, 2013.

Sanchez, M., Fryazinov, O., Adzhiev, V. and Pasko, A., 2014. Space-Time Transfinite Interpolation of Volumetric Material Properties, *IEEE Transactions on Visualization and Computer Graphics*, 2014, Pages 278-288.

Fryazinov, O., Sanchez, M., and Pasko, A., 2014. Shape Conforming Volumetric Interpolation with Interior Distances, *Computer & Graphics*, 2015, Pages 149-155.

Sanchez, M., Fryazinov, O., Fayolle, P.-A., and Pasko, A., 2015. Convolution Filtering of Continuous Signed Distance Fields for Polygonal

Meshes, *Computer Graphics Forum*, 2015.

L. McLoughlin, O. Fryazinov, M. Moseley, M. Sanchez, V. Adzhiev, P. Comninos, A. Pasko, SHIVA: virtual sculpting and 3D printing for disabled children, *Proceedings of the 22nd International Conference on Computers in Education* (November 30- December 4, Nara, Japan), Liu, C.-C. et al. (Eds.), Asia-Pacific Society for Computers in Education, ISBN 978-4-9908014-1-0, 2014, Pages 665-670.