Hybrid Modelling of Heterogeneous Volumetric Objects

Alexander Tereshin

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

The National Centre for Computer Animation

Bournemouth University

December, 2020

Copyright statement

Contents

List of Figures

Tables

List of Acronyms

ADF	- Adaptively Sampled Distance Field
AMF	Additive Manufacturing File
API	- Application Programming Interface
BRep	- Boundary Representation
BVH	- Bounding Volume Hierarchy
CAD	- Computer-Aided Design
CDT	- Chamfer Distance Transform
CPU	- Central Processing Unit
CSG	- Constructive Solid Geometry
DAG	- Directed Acyclic Graph
DF	- Distance Function
DT	- Distance Transform
EVDT	- Efficient Vector Distance Transform
FAV	- Fabricatable Voxel
FGM	- Functionally Grading Material
FIM	- Fast Iterative Method
FMM	- Fast Marching Method
FRep	- Function Representation
FSM	- Fast Sweeping Method
GO-FIM	- Group-Ordered Fast Iterative Method
GPU	- Graphical Processing Unit
HFIM	- Hierarchical Fast Iterative Method
HFRep	- Hybrid Function Representation
HSTB	- Heterogeneous Space-Time Blending
IDF	- Interior Distance Field

NURBS	- Non-Uniform Rational B-Spline
OMT	- Optimal Mass Transport
PDE	- Partial Differential Equations
PHT	- Polynomial splines over Hierarchical T-meshes
RBF	- Radial Basis Function
SARDF	- Signed Approximate Distance Transform
SDF	- Signed Distance Field
SDT	- Signed Distance Transform
SEDT	- Sequential Euclidean Distance Transform
SSEDT	- Signed Sequential Euclidean Distance Transform
SIMD	- Single Instruction, Multiple Data
STB	- Space-Time Blending
STTI	- Space-Time Transfinite Interpolation
STL	- Standard Tesselation Language
TVD	- Total Variation Diminishing
UDF	- Unsigned Distance Field
VCVDT	- Vector-City Vector Distance Transform
VDT	- Vector Distance Transform
VFX	- Visual Effects
3MF	- 3D Manufacturing Format

Abstract

Heterogeneous multi-material volumetric modelling is an emerging and rapidly developing field. A Heterogeneous object is a volumetric object with interior structure where different physically-based attributes are defined. The attributes can be of different nature: material distributions, density, microstructures, optical properties and others. Heterogeneous objects are widely used where the presence of the interior structures is an important part of the model. Computer-aided design (CAD), additive manufacturing, physical simulations, visual effects, medical visualisation and computer art are examples of such applications. In particular, digital fabrication employing multi-material 3D printing techniques is becoming omnipresent. However, the specific methods and tools for representation, modelling, rendering, animation and fabrication of multi-material volumetric objects with attributes are only starting to emerge. The need for adequate unifying theoretical and practical framework has been obvious.

Developing adequate representational schemes for heterogeneous objects is in the core of research in this area. The most widely used representations for defining heterogeneous objects are boundary representation, distance-based representations, function representation and voxels. These representations work well for modelling homogeneous (solid) objects but they all have significant drawbacks when dealing with heterogeneous objects. In particular, boundary representation, while maintaining its prevailing role in computer graphics and geometric modelling, is not inherently natural for dealing with heterogeneous objects especially in the context of additive manufacturing and 3D printing, where multi-material properties are paramount as well as in physical simulation where the exact representation rather than an approximate one can be important.

In this thesis, we introduce and systematically describe a theoretical and practical framework for modelling volumetric heterogeneous objects on the basis of a novel unifying functionally-based hybrid representation called HFRep. It is based on the function representation (FRep) and several distance-based representations, namely signed distance fields (SDFs), adaptively sampled distance fields (ADFs) and interior distance fields (IDFs). It embraces advantages and circumvents disadvantages of the initial representations. A mathematically substantiated theoretical description of the HFRep with an emphasis on defining functions for HFRep objects' geometry and attributes is provided. This mathematical framework serves as the basis for developing efficient algorithms for the generation of HFRep objects taking into account both their geometry and attributes.

To make the proposed approach practical, a detailed description of efficient algorithmic procedures has been developed. This has required employing a number of novel techniques of different nature, separately and in combination. In particular, an extension of a fast iterative method (FIM) for numerical solving of the eikonal equation on hierarchical grids was developed. This allowed for efficient computation of smooth distance-based attributes.

To prove the concept, the main elements of the framework have been implemented and used in several applications of different nature. It was experimentally shown that the developed methods and tools can be used for generating objects with complex interior structure, e.g. microstructures, and different attributes. A special consideration has been devoted to applications of dynamic nature. A novel concept of heterogeneous space-time blending (HSTB) method with an automatic control for metamorphosis of heterogeneous objects with textures, both in 2D and 3D, has been introduced, algorithmised and implemented. We have applied the HSTB in the context of '4D Cubism' project. There are plans to use the developed methods and tools for many other applications.

Key words: *heterogeneous volumetric objects*, *scalar fields*, *function representation. distance fields*, *multi-materials*, C^1 *continuity*, *hybrid representation*, *time-variant objects*

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Valery Adzhiev for his continuous support, humour and advice, and introduction into functionally-based modelling and modern art. Thank you for the long discussions of theoretical aspects of my work and for being very helpful on this long and interesting journey.

I would like to express my gratitude to my another supervisor Prof. Alexander Pasko for being very helpful with working on theoretical solutions and interesting discussions of my topic as well as for introducing me to our working team.

I would like express my gratitude to my third supervisor Dr. Oleg Fryazinov for helping me with solving technical and mathematical problems related to my thesis and sharing his thoughts on some algorithmic parts of my work.

I would like to say thank you to Dr. Eike Anderson who helped me with editing of my two papers and share some insights related to colour interpolation in different colour spaces.

I am very grateful for the help provided by my faculty, especially by Sonia Ashby. She always helps with finding a solution for any problem not related to my research and her support was really great.

I am very grateful having such friends as Mara Catalina Aguilera Canon, Miguel Ramos Carretero, Larisa Boruzdina, Vasily Burdonsky, Alexandra Belskaya, Mariia Lopukhova and many others for supporting me during this complicated journey into the world of function-based modelling.

Finally, I would like to say thank you to all my family, especially to my father, uncle, sister and cousin for their believe in me and support that makes my journey a bit easier.

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 materials related to hybrid function representation were initially published in (Tereshin, Adzhiev, Fryazinov, and Pasko 2019). Then the theoretical framework was extended and a full-scale paper was published in Graphical Models journal (Tereshin, Pasko, et al. 2021). The corresponding preprint (Tereshin, Pasko, et al. 2020) was published on arXive. The paper describing the method for handling automatically controlled 2D metamorphosis between textured objects was published in SIAM Journal of Imaging Sciences (Tereshin, Adzhiev, Fryazinov, Marrington-Reeve, et al. 2020). Finally, the concept of the heterogeneous space-time blending was introduced in (Tereshin, Anderson, et al. 2020).

Chapter 1

Introduction

There is a wide range of natural and man-made objects in the physical world. The prevailing conventional computer technology successfully deals with homogeneous objects made of a uniform material. There are a number of well-developed theoretical and practical methods for modelling, rendering, animating and fabricating such objects. In particular, several well-established and mathematically substantiated representational schemes has been developed for those objects.

However, the majority of objects, be they natural or digitally created, are not homogeneous. They can be composed of different materials and have a complex interior structure with associated physical properties. Such objects are called heterogeneous objects. This is a rather broad notion. In this work, we focus on developing a theoretical modelling framework and practical methods to deal with heterogeneous objects considering their geometric shape in concert with the internal attributes representing the object's properties. This will be beneficial for many existing and emerging applications in such areas as CAD modelling, additive manufacturing, multi-material design, bio-engineering, medical research, and others.

There will be a particular focus on dynamic (time-variant) heterogeneous objects. The theoretical and practical aspects of dealing with that kind of objects are not properly developed. They are very important in such modern applications as physical simulations, computer animation and visual effects (VFX).

In recent years, 3D printing technologies have become omnipresent across different industries and applications. The resolution capabilities of the modern 3D printers is relatively high. Therefore, it is possible to reproduce an object with a high level of detail including various multi-scale micro-structures in interior of the object. Modern 3D printers are able to print with multiple materials that can be mixed with each other. The modelling framework for heterogeneous objects should be seamlessly matched with the technical aspects of up-to-date 3D printing.

Geometric shape of the heterogeneous object can be defined using different representational schemes that have various advantages and drawbacks. The most widely used representation for geometry definition is the boundary representation (BRep). This representation is not exactly natural for defining objects with interior structures unless some special techniques are applied, e.g. nested surface shells. Nevertheless, this representation maintains its prevailing role due to its numerous well-known advantages. It is widely used in 3D printing as many file formats, including the most popular STL format, assume a BRep object to be processed and printed.

With respect to attributes, heterogeneous objects can be split into two categories: composite objects and functionally graded materials (FGMs) and structures. Composite objects are usually made of two or more constituent materials with significantly different physical properties. on the contrary, FGMs are constructed by gradual variation of the material and structure in the composition over the volume. This is reflected in introducing relevant attributes. This thesis deals with various issues of how multiple attributes can be defined in heterogeneous objects.

Volumetric representations in the form of voxels are more natural for defining such heterogeneous objects as they are based on volumetric grids. Voxels represent an object as a set of cubic cells at which the geometry along with the object attributes are defined. However, this representation essentially approximates both the geometry model and the material distribution in interior of the object as their definition is limited by the resolution of the voxel grid.

on the other hand, function-based, and more specifically, distance-based representations related to scalar fields of different kinds are able to represent the object and its interior structure in both continuous and discrete forms. They are exact, embrace a wide range of geometric shapes and naturally define many physically-based attributes. There are a lot of well-established operations for these representations. Most of them provide distances to the object surface. The distance property is important in scalar field based modelling as it provides a predictable control over the resulting field.

However, distance functions (DFs) are not essentially continuous, they can have medial gradient discontinuities and are not necessarily smooth. This potentially results in non-smooth surfaces. Undesired artefacts, such as creases, can appear after applying some operations, for instance, blending and metamorphosis, which are important for many applications.

As to attribute functions, they can be parameterised by distances to provide an intuitive control over attribute distributions in interior of the object. However, if the distance function is non-smooth, it could lead to undesired artefacts (stresses, creases, etc.) in the defining attributes.

The most common schemes of that type are function representation (FRep), the signed distance function representation (SDF), the adaptively sampled distance function representation (ADF) as well as the shape aware distance fields which are represented by functions that we call interior distance functions (IDF). We consider these function-based and distance-based representations as a promising conceptual and practical schemes to deal with heterogeneous objects, especially in the context of a number of topical application areas concerned with exact volume-based geometric modelling, animation, simulation and fabrication.

However, the existing representational schemes of that type appear in many variations. For instance, SDFs can be obtained using various approximation or exact methods that aim to solve one or several SDF drawbacks. ADFs can be defined using various methods as well, but only some of them are capable to solve the problems introduced in ADF original implementation (Frisken et al. 2000). IDFs unify a group of methods that aim to compute various types of distances on the surface and in interior of the object. Therefore, a single unifying framework that would cover all variations of this type of representations have not been formulated yet. There is an obvious need for a properly substantiated and unifying theoretical and practical framework. This challenge can be considered in the context of the emergence of new representational paradigms suitable for the advanced applications, such as modelling of material structures, that was outlined and substantiated in (Regli et al. 2016).

In this thesis we propose a novel function-based representational scheme. Having considered four representations identified above, we have chosen FRep as a basis representation for constructing a novel hybrid framework that unifies advantages of

FRep, SDFs, ADFs and IDFs and compensate for their drawbacks.

We introduce a mathematical framework called hybrid function representation (HFRep) for defining a heterogeneous volumetric object with its attributes in continuous and discrete forms. This representational scheme aims at dealing with heterogeneous objects with some specific time-variant properties related to both geometry and attributes that are important in physical simulations and animations.

On the basis of the proposed theoretical framework, we have developed an algorithmic procedure allowing to generate HFRep objects in terms of their geometry and attributes. To make the approach practical, we have provided a detailed description of the main steps of the algorithm and identified some problematic issues associated with them. This has required developing some novel techniques and employing a number of existing techniques of different nature, separately and in combination. Among them is an adaptation of the fast iterative solver of the eikonal equation on 2D hierarchical grids that provides the solution which is continuous and smooth.

The proposed theoretical and algorithmic framework can be used for generating various heterogeneous objects consisting of geometric and attribute parts that can be considered in the context of CAD modelling, 3D manufacturing or artistic applications. The generated objects can be time-dependent. Therefore, such objects can be used in dynamic simulations or applications where the presence of the interior structure with specified attributes is important to take into account.

To show how the proposed approach works in the context of different applications, we have implemented several case-studies including those that deal with colour, material and microstructure attributes in the interior of functionally-defined shapes in the context of time-variant modelling. In particular, a novel heterogeneous space-time blending technique allowing to handle geometric and attribute transformations simultaneously and interconnectedly. This method has been applied to the heterogeneous objects of artistic nature, in particular within '4D Cubism' project.

1.1 Research Problems

In this thesis we are going to explore theoretical, algorithmic and practical research problems concerned with exploring and developing a novel representation for heterogeneous objects. These research problems are formulated as follows:

1. In the current literature, the existing volumetric and function-based representational schemes, potentially suitable for heterogeneous objects, appear in many variations and the field as a whole exhibits a rather fragmented suite of methods. A thorough survey of the relevant function-based representations is needed. on the basis of that survey, it will be possible to suggest a proper classification of the existing representations as well as to identify their advantages and drawbacks.

2. Function-based and distance-based representations are considered as a promising way to deal with heterogeneous objects. These representations define a volume as a continuous scalar field with the field resolution bounded by the computational precision. This type of volumetric representations allows for defining attributes along with geometric shape of the object that is important in the context of heterogeneous volumetric modelling. There is a need for a properly substantiated and unifying theoretical and practical representational framework. This hybrid framework should be mathematically rigorous and systematically described. The main functionality and properties of

this framework should be developed. The objects, operations, and relations should make the hybrid framework self-contained in terms of heterogeneous object modelling.

3. More specifically, it is essential to investigate how the existing function-based representations, related to scalar fields of different kinds, can be combined together in a new hybrid representation for heterogeneous objects. The combination of two or more representations together will allow to inherit their advantages and compensate for their drawbacks. One needs to explore whether it is possible to construct the heterogeneous objects' defining functions (that should be in the core of the mathematical representational framework) that are continuous and have a distance property everywhere in a Euclidean space. If the defining function is at least one time differentiable, then it is smooth. Finally, the distance property will provide a consistent and predictable way for defining the geometric shape of the object as well as its attributes.

4. The theoretical framework should be supported by the algorithmic framework. The basic algorithm, allowing to generate heterogeneous objects in terms of their geometry and attributes, should be in the core of it. This algorithm should be developed in detailed step-by-step manner and will probably need to be supported by development of a number of specific techniques guaranteeing that the defining functions being constructed meet the framework's requirements.

5. It is essential to investigate how various attributes (e.g. colour, multi-materials, microstructures) can be defined in the context of the new hybrid framework for heterogeneous object modelling. It is important to identify how to establish correspondences between defined geometry and attributes to provide an intuitive control over their definition necessary for practical work with it.

6. It is essential to investigate how heterogeneous objects defined by the new hybrid framework can be considered in the context of the dynamic applications, taking into account time-variant geometry and attribute transformations that should happen in an interconnected manner.

7. Finally, it is important to explore whether the proposed framework is suitable to be used in several modern applications. Artistic applications with dynamic flavour can be of particular interest.

1.2 Solution Statement

A number of original solutions have been achieved in the process of exploring and solving the research problems presented in the previous section.

1. Four conventional representational schemes related to scalar fields of different kinds, namely function representation (FRep), the signed distance function representation (SDF), the adaptively sampled distance function representation (ADF) and the interior distance function representation (IDF) have been identified along with their advantages and drawbacks. A formalisation of the notions of FReps, ADFs and IDFs has been proposed.

2. A mathematically substantiated modelling framework for heterogeneous objects called hybrid function representation (HFRep) that is based on the combination of FRep with one of the three distance-based representations, namely SDF, ADF and IDF, has been introduced. The mathematical properties of the HFRep function have been rigorously described. The properties of the supported HFRep heterogeneous objects have been systematically described, and the operations over HFRep objects, both those that preserve Euclidean distances and those that do not, were identified.

3. The basic algorithm for generating HFRep objects in terms of their geometry and attributes with taking into account the type of hybridisation has been proposed.

4. The HFRep algorithmic framework that includes a detailed description of all the steps of the basic algorithm for generating different types of distance fields and their bi-pair hybridisation with FRep representation has been developed. Several techniques allowing for constructing the FRep defining function with particular properties have been suggested to use, both separately and in combination.

5. A novel algorithmic solution for computing the signed vector-city distance transform was developed. The hierarchical fast iterative method (HFIM) method has been suggested and implemented in the form of a step-by-step algorithm. The method provides at least C^1 continuous unsigned distance field that is restored at each cell of the grid using the PHT-splines. The solution of the eikonal equation is computed using a modified version of the first order Godunov upwind discretisation scheme for computations on the hierarchical grids.

6. Several methods for defining various attributes (colour, microstructures, materials) in the interior of the 2D and 3D HFRep object have been suggested.

7. A novel method allowing for automatically controlled morphing between two topologically arbitrary 2D shapes with sophisticated textures has been developed. The method allows for a smooth transition between source and target objects (considered as HFRep objects) by generating in-between shapes and associated textures without setting any correspondences between boundary points or features.

8. A novel heterogeneous space-time blending (HSTB) method that can handle holistic heterogeneous objects, rather than their geometric and attribute components in separation, especially in the context of automatically controlled metamorphosis between topologically arbitrary 3D and 4D textured objects, has been proposed. The basic algorithm and several techniques, extended to 3D, that solve the identified drawbacks, have been developed. The HSTB method was technically implemented as the custom HSTB node *SideFX Houdini* and was applied to deal with dynamic (time-variant) textured objects in the context of an artistic project '4D Cubism'.

1.3 Thesis Structure

This thesis is structured as outlined below.

In chapter 2, the related works relevant to the topic of the research are discussed. Several core representations for defining heterogeneous objects, namely boundary representation, voxel representation as well as several function-based and hybrid representations are reviewed. As heterogeneous objects consist of geometric and attribute parts, state-of-the-art methods for defining attribute distribution in volumes are discussed as well as how such objects can be considered in the context of dynamic applications where the presence of the interior structure is important.

In chapter 3, the theoretical description of the introduced hybrid framework is presented. This chapter covers important mathematical definitions that will serve as the basis for constructing the hybrid framework. The definitions of the function representation, adaptively sampled distance field and interior distance field are given in this chapter. Important mathematical properties of the proposed hybrid framework as well as the basic algorithm for its generation is presented. The theoretical framework described in this chapter was first presented on Eurographics 2019 (Tereshin, Adzhiev, Fryazinov, and Pasko 2019) and then published in paper 'Hybrid function representation for heterogeneous objects' (Tereshin, Pasko, et al. 2020, 2021).

In chapter 4, the algorithmic framework for the hybrid representation is discussed in details. This chapter covers the main steps of the basic algorithm for the HFRep generation introduced in the previous chapter. In this chapter a detailed description of the extended fast iterative method (hierarchical fast iterative method) proposed in this work is presented. At the end of this chapter the implementation of the proposed hybrid framework and the obtained results are discussed. The adaptation of the fast iterative method for the computation of the solution of the eikonal equation on hierarchical grids as well as algorithmic implementation of HFRep were published in paper 'Hybrid function representation for heterogeneous objects' (Tereshin, Pasko, et al. 2020, 2021).

The next two chapters 5 and 6 show how the proposed hybrid framework can be used in two particular applications that deal with morphing of one object into another one.

In chapter 5, a 2D automatically controlled metamorphosis without establishing any correspondences between textured shapes or shapes defined by distance-based functions with specified attribute distributions is described. The proposed method is based on the combination of two well-established methods, namely space-time blending and space-time transfinite interpolation. Several important mathematically substantiated techniques for enhancing the results of the space-time blending are presented. Finally, applications and the obtained results are discussed. The description of the proposed framework presented in this chapter was published in paper 'Automatically Controlled Morphing of 2D Shapes with Textures' (Tereshin, Adzhiev, Fryazinov, Marrington-Reeve, et al. 2020).

In chapter 6, an extended space-time blending method for 3D heterogeneous objects is proposed. This method allows to compute geometry and attribute transformations simultaneously and interconnectedly. The introduced method is used in an artistic application called '4D Cubism' and the results are presented at the end of this chapter. This method was presented on Eurographics 2020 (Tereshin, Anderson, et al. 2020).

In chapter 7, a summary of problems and solutions presented in this thesis is given. The chapter concludes by outlining both the main contributions of the work and the future research directions.

Chapter 2

Related Work

In this chapter we review different representations and methods for defining heterogeneous objects. The heterogeneous object consists of two parts: geometric shape and attributes. Therefore, we start this chapter from reviewing of the representations that are used to define the geometric shape of the heterogeneous object. We start our literature review from the well-established boundary representation (BRep) that still prevails in industry. Then we describe the voxel representation that is widely used in volumetric modelling as well as in the context of the dynamic volumetric simulations. After that, we describe the scalar field-based representations, namely, implicit surfaces, FReps, RBFs, SDFs, ADFs, IDFs and various hybrid representations. We identify their advantages, drawbacks and made a comparative analysis. Then we review various approaches for defining different attributes in objects interior.

As heterogeneous objects have interior structure, they are particularly useful for the accurate dynamic simulations and dynamic transformations (e.g. morphing, metamorphosis, etc.). In this work we are interested in time-variant objects that can be used in such applications as morphing or metamorphosis. Therefore, we review state-of-the-art methods for 2D and 3D morphing and metamorphosis operations.

At the end of this chapter we cover the core rendering techniques for heterogeneous objects with various specified attributes.

2.1 Heterogeneous Objects

Heterogeneous volumetric object modelling is a rapidly developing field that has a variety of different applications. Volumetric modelling is concerned with computer representations of object surface geometry as well as its interior. Homogeneous volume modelling, better known as solid modelling, deals with volume interior uniformly filled by a single material. Heterogeneous object is a volumetric object with geometric shape that is considered in concert with its internal attributes defined for each point of the shape. These attributes represent such physical properties as material, density, colour and others (Kou and Tan 2007; Li, Fu, et al. 2020). Let us give a formal definition of the heterogeneous object:

Definition 2.1.1. *Let the object O_H be defined as a two component tuple: geometric shape $G \subseteq X$ in the form of a multidimensional point-set geometry, where $X \subset \mathbb{R}^n$, and attributes A_i corresponding to the physical and material properties of the object O_H. Then such an object O_H is a heterogeneous object defined as:*

$$O_H := (G, A_1, ..., A_n), \tag{2.1}$$

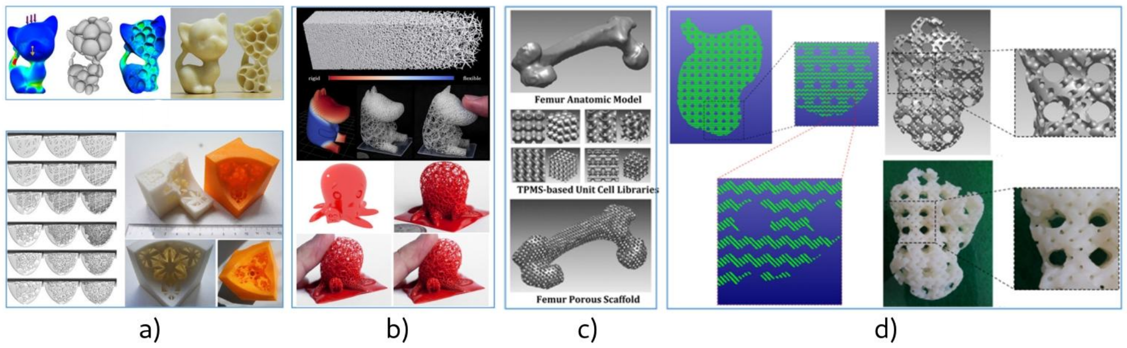


Figure 2.1: *Examples of heterogeneous objects with various interior structure. a) Various cellular structures. b) Voronoi-based foam structures. c) Various porous scaffolds. d) Distance-based hierarchical scaffolds. Reprinted from (Li, Fu, et al. 2020).*[a]*.*

[a]Creative Commons Attribution 4.0 International License (`http://creativecommons.org/licenses.by/4.0/`), two images from the paper were combined in one).

where $n \in \mathbb{N}$ is the number of attributes.

This type of objects is widely used in applications where the presence of the interior structures is an important part of the model. Additive manufacturing, physical simulation and visual effects are examples of such applications.

In Fig. 2.1 we show several examples of heterogeneous objects. The geometric shape G of the heterogeneous object consists of the surface and interior structure. As it follows from this figure, the interior structure can be defined as cellular structures (see Fig. 2.1, a), Voronoi-based foam structures (see Fig. 2.1, b), various porous scaffolds (see Fig. 2.1, c, d). Various attributes A_i in interior of such objects can be presented using distribution functions.

The most widely used representations for defining heterogeneous objects are boundary representation, distance-based representations, function representation and voxels. Boundary representation (BRep) (Lei et al. 2014) maintains its prevailing role due to its numerous well-known advantages. It works well in solid modelling for objects consisting of a set of polygonal surface patches stitched together to envelope the uniform and homogeneous structure of its material. However, BRep is not inherently natural for dealing with heterogeneous objects, especially in the context of additive manufacturing and 3D printing (Livesu et al. 2017), where volume-based multi-material properties are paramount as well as in physical simulation where the exact representation rather than an approximate one can be important (Nealen et al. 2006).

On the contrary, volumetric representations in the form of voxels (Wang et al. 2011) are more natural for defining such heterogeneous objects as they are based on volumetric grids. Voxels represent an object as a set of cubic cells at which the geometry along with the object attributes are defined. However, this representation essentially approximates both the geometry model and the material distribution in interior of the object, as their definition is limited by the resolution of the voxel grid.

On the other hand, function-based, and more specifically, distance-based representations are able to represent the object and its interior structure in both continuous and discrete forms (Jones, Baerentzen, and Sramek 2006).

2.2 Boundary Representation

The boundary representation (BRep) (Muuss and Butler 1991) defines the object as a set of connected surface elements. BRep consists of topological and geometrical parts (surfaces, curves and points). The topology of the BRep object consists of faces, edges and vertices. A geometrical part of the BRep object contains the description of the points that belong to the surface of this object. BRep has a number of well-established operations and can be defined using two well-developed representational schemes, namely, parametric surfaces and polygonal meshes. In this section we review the main advantages and drawbacks of BRep in the context of the heterogeneous object modelling.

2.2.1 Parametric Representation

A parametric representation is defined as a mapping from the parametric space into a possibly higher dimensional space

$$\boldsymbol{F}(t) : \mathbb{R}^m \rightarrow \mathbb{R}^n; \quad \boldsymbol{F}(t) = (f_1(t), ..., f_n(t)); \quad t \in [a, b] \subset \mathbb{R}^m \tag{2.2}$$

where $\boldsymbol{F}(t)$ is a parametric object that is defined as a set of functions $f_n(t)$ parameterised by $t \in [a, b]$. This approach is widely used for defining complex curves or surfaces

In most cases, a parametric object representation is based on polynomials that are efficient to compute. For example, a parametric surface can be constructed using two parametric curves $C_1(u) = \sum_{i=1}^{n} f_i(u)P_{1,i}$ and $C_2(v) = \sum_{j=1}^{m} g_j(v)P_{2,j}$ in u and v directions, where $P_{1,i}$ and $P_{2,i}$ can be points or vectors. The resulting function can be defined as:

$$\boldsymbol{F}(u, v) = \sum_{i=1}^{n}\sum_{j=1}^{m} f_i(u)g_j(v)\boldsymbol{P}_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{m} b_{ij}(u, v)\boldsymbol{P}_{ij} \tag{2.3}$$

where $b_{ij}(u, v)$ are basis functions, u and v are independent parameters and $\boldsymbol{P}_{ij}$ can be defined as points or vectors. Parametric surfaces usually consist of surface patches $\boldsymbol{F}(u, v)$ which are defined by control points $\boldsymbol{P}_{ij}$. However, parametric surfaces do not necessary go through these control points.

There are two basic types of parametric surfaces which are widely used in 3D modelling: rational Bezíer surfaces (Piegl and Tiller 1997) and B-spline surfaces in the form of the non-uniform rational B-splines (NURBS) (Versprille 1975). Bezíer surfaces are represented with quadrilateral patches that are interpolating its corner points. At each corner of the patch, which lies within the convex hull of its control vertices, the tangent plane interpolates the corner vertex and the two neighbouring edge vertices. The rational Bezíer patch can be defined as

$$\boldsymbol{F}(u, v) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} \omega_{ij}\boldsymbol{P}_{ij}B_i^n(u)B_j^m(v)}{\sum_{i=1}^{n}\sum_{j=1}^{m} \omega_{ij}B_i^n(u)B_j^m(v)} = \sum_{i=1}^{n}\sum_{j=1}^{m} b_{ij}(u, v)\boldsymbol{P}_{ij} \tag{2.4}$$

where ω_{ij} are non-negative weights, the $\boldsymbol{P}_{ij}$ forms the control polygon, $B_i^m(u), B_j^n(v)$ are Bernstein polynomials, $b_{ij}(u, v)$ are basis functions, and u, v are parameters.

Bezíer surfaces are continuous and require fewer points to represent curved surfaces. However, Bezíer patch meshes are difficult to render directly. It is complicated

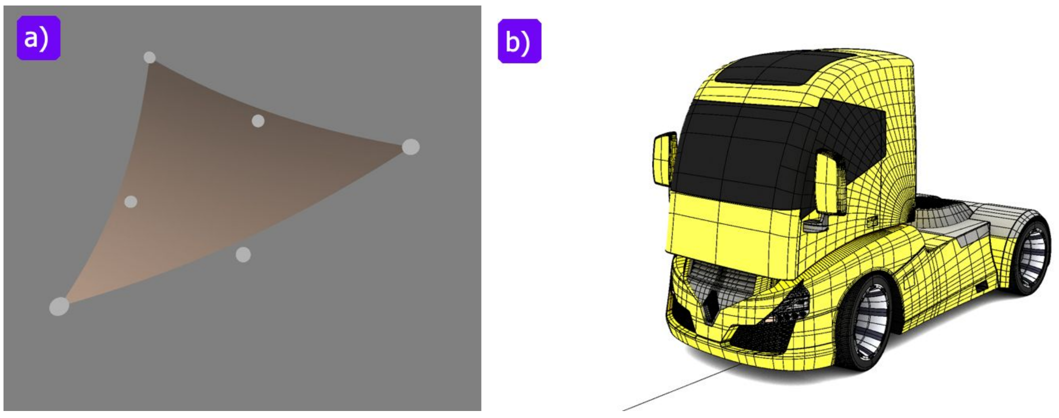


Figure 2.2: *Examples of the NURBS surfaces. a) A Bezíer triangle patch. b) A concept of the truck (NURBS model)* [a].

[a]The truck model was used under Royalty Free License (https://www.cgtrader.com/free-3d-models/vehicle/truck/renault-radiance).

to compute intersection operations with lines and to combine the meshes directly with the perspective projection algorithms (Farin 2002).

On the other hand, NURBS surfaces (see Fig. 2.2, b) are more convenient to use. NURBS surfaces consist of a small amount of patches. They provide the flexibility to design a large variety of shapes, and they provide one common mathematical form for standard analytical shapes and free-form shapes. NURBS can be defined in general case as

$$\boldsymbol{F}(u,v) = \frac{\sum_i^n \sum_j^m \omega_{ij} \boldsymbol{D}_{ij} N_i^n(u) N_j^m(v)}{\sum_i^n \sum_j^m \omega_{ij} N_i^n(u) N_j^m(v)} = \sum_i^n \sum_j^m b_{ij}(u,v) \boldsymbol{D}_{ij} \tag{2.5}$$

where ω_i are non-negative weights, the $\boldsymbol{D}_{ij}$ are control points, $N_i^n(u), N_j^m(v)$ are B-spline basis functions, $b_{ij}(u,v)$ are basis functions and u, v are parameters.

The NURBS surfaces are invariant under affine transformations. They can be used for designing a large variety of shapes and can be evaluated reasonably quickly using numerically stable methods (Farin 1996; Plegl and Tiller 1995). The NURBS curve or surface can be split into several rational Bezíer surfaces of the same degree. These patches can be used for creating complex objects by stitching them together.

One of the important properties of the parametric surfaces is *geometric continuity*. Formally, it can be defined for curves and splines as follows (Kiciak 2016):

Definition 2.2.1. *(Geometric continuity, curves) A curve has a geometric continuity G^n of order $n \geq 1$ if there exists a local regular parametrisation of this curve that is C^n continuous in a neighbourhood of each point of this curve.*

Definition 2.2.2. *(Geometric continuity, surfaces) A surface has a geometric continuity G^n of order $n \geq 1$ if there exists a local regular parametrisation of this surface that is C^n continuous in a neighbourhood of each point of this surface.*

The existence and continuity of the function derivatives guarantees the smoothness of its graph. However, when we are dealing with parametric representations (e.g Bézier curves/surfaces, NURBS), the continuity of derivatives of any parametrisation does not guarantee the smoothness of the curve or surface. Therefore, the notion of the regularity of the parametrisation is introduced in both definitions

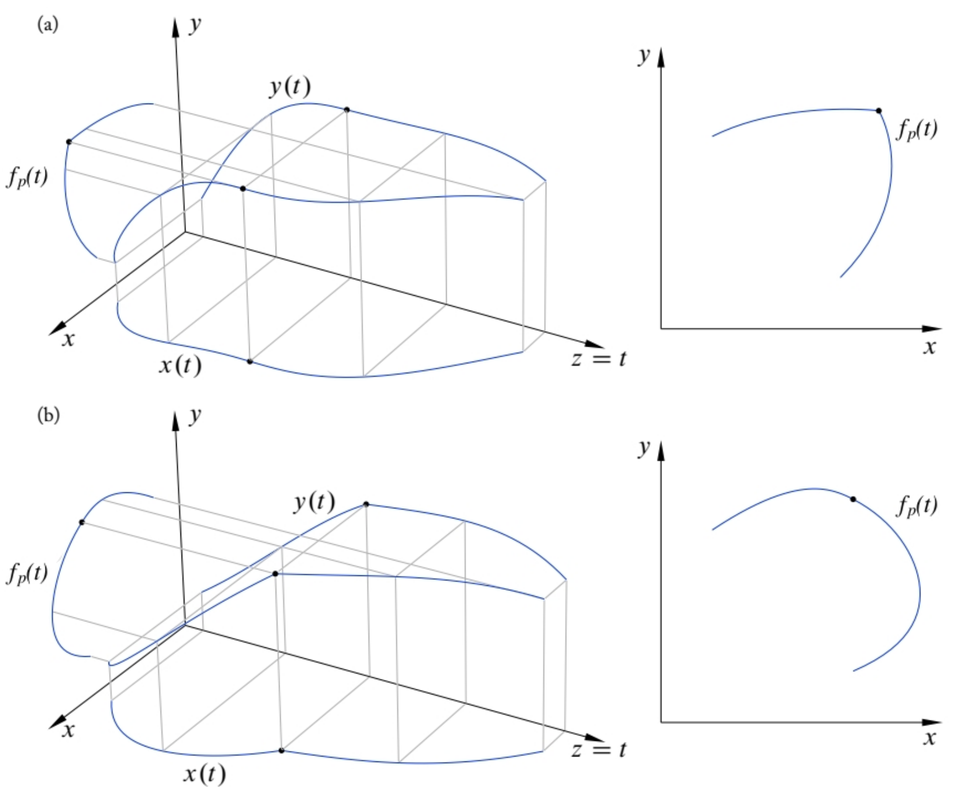


Figure 2.3: *A comparative picture of two types of continuity: parametric and geometric. a) a non-smooth curve with a C^2 continuous parametrisation; b) a curve with G^2 geometric continuity that has a parametrisation with discontinuous derivatives. Reprinted from (Kiciak 2016)), Copyright ©2016, with permission from Morgan & Claypool Publishers.*

above. A vector function $\boldsymbol{f}(x_0, ..., x_n)$ of n variables x is regular at a point $\boldsymbol{p}$ if its partial derivatives at $\boldsymbol{p}$ are linearly independent vectors (Kiciak 2016).

In Fig. 2.3 two planar curves with their parametrisation are shown. The graph of the parametrisation $f_p(t)$ consists of two functions $x(t)$ and $y(t)$ and its three-dimensional curve is obtained using the identity function $z(t) = t$. Both functions $x(t)$ and $y(t)$ in Fig.2.3 (a) have continuous derivatives up to second order. However, the curve has a point of discontinuity, where curves defined by these functions are stitched together. On the contrary, the parametrisation of the curve in Fig. 2.3 (b) has discontinuous derivatives at some points, but the curve is smooth at each point (i.e. the tangent line exists) and its curvature is continuous.

Let us assume that a regular parametrisation $f_p(t), t \in [a, t_0]$ is given and it is essential to construct a parametrisation $f^*_p(u), u \in [u_0, b]$ that is C^n continuous and its arcs defined by two parametrisations form a curve with geometric continuity G^n. Let $t = f(u)$, where $f(u)$ is a monotone and at least C^n continuous and regular function. If $f(u_0) = t_0$, then piecewise parametrisation will be defined on the domain $[f^{-1}(a), b]$.

If we would like to obtain a curve with G^1 geometric continuity, the following interpolation conditions for the parametrisation $f^*_p(t)$ should be satisfied:

$$f^*_p(u_0) = f_p(t_0); \quad f^{*\prime}_p(u_0) = t_1 f'_p(t_0); \quad t_1 = f'(u_0). \tag{2.6}$$

To obtain a curve with G^2 geometric continuity, the following condition should be

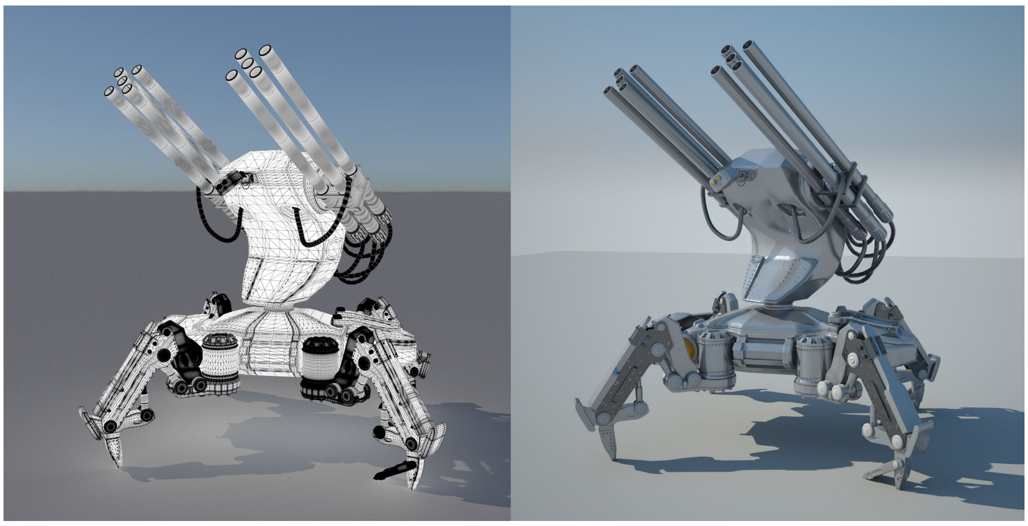


Figure 2.4: *Example of the high resolution polygonal mesh of the robot (right image) with its wireframe (left image) (the BRep model was done in Autodesk Maya by the author).*

also satisfied:

$$f_p^{*\prime}(u_0) = t_2 f_p(t_0) + t_1^2 f_p''(t_0) \tag{2.7}$$

Finally, the curve with G^3 geometric continuity should also satisfy this equation:

$$f_p^{*\prime\prime\prime}(u_0) = t_3 f_p'(t_0) + 3t_1 t_2 f_p''(t_0) + t_1^3 f_p'''(t_0) \tag{2.8}$$

Unfortunately, NURBS and Bezíer surfaces have a lot of drawbacks. The lack of a proper geometric continuity can be crucial if a stitching operation between two parametric surfaces is applied. It is difficult to compute intersections between parametric surfaces or to detect self-intersections in them. To resolve these issues different numerical solutions are applied (Guthe, Balázs, and Klein 2005; Wang 1996). Some of them deal with solving partial differential equations (PDEs). Some others use numerically time-consuming approaches that, first, subdivide the surface, then intersect them and, finally, refine the obtained result (Houghton et al. 1985).

2.2.2 Polygonal Representation

Formally, a polygon is defined as a closed region of the plane bounded by a finite number of line segments stitched together to form a closed, non self-intersecting closed curve (Satyan and O'Rourke 2011). A polygonal mesh (M_p) (see Fig. 2.4) is a set of vertices (V), edges (E) and faces (F) that defines the shape of a polyhedral object (Hughes et al. 2014):

$$\begin{aligned} M_p &= \{V, E, F\} \\ V &= \{v_1, ..., v_n\}; \\ E &= \{e_1, ..., e_l\}; \quad e_l = \{v_j, v_k\}; \quad 0 < j, k \leq n \\ F &= \{f_1, ..., f_m\}; \quad f_i = \{e_j, e_k, ..., e_p\}; \quad 0 < j, k, p \leq l; \end{aligned} \tag{2.9}$$

This representation defines the object in the form of a shell, where the surface of the object can be formed with triangles, quadrilaterals or other simple convex

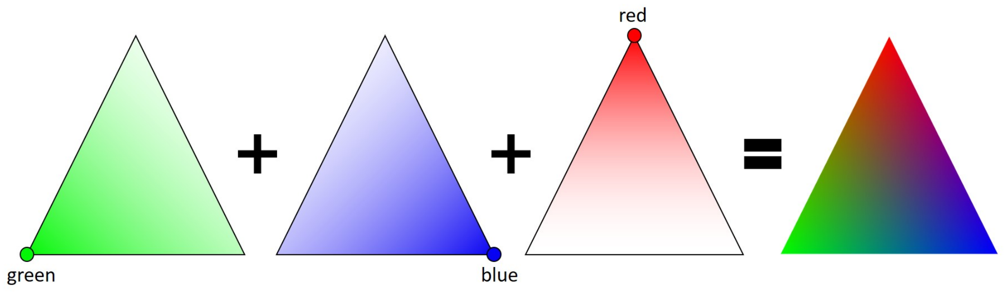


Figure 2.5: *Shading of the triangle using barycentric interpolatiob.*

polygons. One of the most widely used simplexes that can be used for approximating the surface are triangles. The formal definition for the triangle meshes can be given as follows (Giblin 2010):

Definition 2.2.3. *A closed surface is a collection M_{tr} of triangles in some Euclidean space that:*

1. *M_{tr} satisfies the intersection condition: two triangles are disjoint, or have one or two vertices in common;*
2. *M_{tr} is connected;*
3. *every vertex v of a triangle of M_{tr} is connected through a simple closed polygon.*

Polygon meshes are widely used in many areas dealing with computer graphics and geometric modelling, such as game development, film and animation production, architecture, CAD modelling and others. There are a lot of well-established operations for polygonal meshes like set-theoretic (Yongbin et al. 2009), smoothing (Botsch et al. 2007), simplification (Heckbert and Garland 1997) and others. Polygonal meshes are widely supported by the modern graphics hardware. Therefore, they can be rendered using various algorithms and techniques in a fast and convenient way.

Even if the geometry is approximated by a polygon grid, it is possible to obtain a continuous distribution of some parameters on the surface of this object using barycentric interpolation (Hormann 2014). For instance, in Fig. 2.5 we show an example of application of the barycentric interpolation to colours defined in vertices of the triangle. If the object is tetrahedralised (i.e. formed using tetrahedral primitives), it is possible to continuously interpolate the data in interior of such a mesh using barycentric interpolation.

The barycentric interpolation is based on the concept of barycentres or centres of mass. Let us consider a system of $n+1$ particles with corresponding coordinates $\boldsymbol{p}_0, ..., \boldsymbol{p}_n$ and masses $w_0, ..., w_n$. In this case, the barycentre of this particle system is the unique point $\boldsymbol{p}$ that satisfies

$$\sum_{i=0}^{n} w_i(\boldsymbol{p} - \boldsymbol{p}_i) = 0 \quad \boldsymbol{p} = \frac{\sum_{i=0}^{n} w_i \boldsymbol{p}_i}{\sum_{i=0}^{n} w_i} \tag{2.10}$$

where w_i are the barycentric coordinates of $\boldsymbol{p}$ with respect to $\boldsymbol{p}_i$. If we consider data $f_0, ..., f_n$ that can be obtained using some function $f(\boldsymbol{p}_i) : \mathbb{R}^m \mapsto \mathbb{R}$. Then the

Rep.	Advantages	Drawbacks
Parametric	• Continuous surface representation; • Bezier surfaces require fewer points to define curved surface; • NURBS provide one common mathematical form for standard analytical shapes and free-form shapes; • NURBS are invariant under affine transformations; • Supports interactive shape design;	• Problematic to calculate intersections between parametric surfaces; • Problematic to resolve self-intersections; • Problematic to render without conversion to polygonal representation; • Lack of geometric continuity;
Polygonal	• A lot of different operations have been already developed; • Efficient to use in different simulation-based and animation-based techniques, where the goal is to deform the boundary surface of the object; • Efficient to render using different physically-based techniques; • Supports interactive shape design;	• Memory inefficient in terms of data storing; • Approximated (discrete) representation of the object; • Suffers from self-intersections, holes and cracks and other inconsistencies; • Object is defined as a polygonal shell that results in problems with defining heterogeneous objects;

Table 2.1: *Advantages and drawbacks of the boundary representation.*

barycentric interpolation for this data is defined as

$$F(\boldsymbol{p}) = \sum_{i=0}^{n} F_{bar_i}(\boldsymbol{p}) f_i \tag{2.11}$$

where $F_{bar_i}(\boldsymbol{p})$ are barycentric basis functions. Some of them can be found in work (Hormann 2014). We will need this notion in chapter 4 for implementing one of the methods.

Polygon meshes can be used for creating implicit surfaces (Yngve and Turk 2002). In this work the authors used implicit surfaces based on variational interpolation technique. They created an iterative method which added new constraints to a variational implicit representation in an iterative manner.

However, set-theoretic operations over polygonal meshes can suffer from self-intersections. Polygonal meshes are also not suitable for operations that change the topology of the object, such as morphing.

One of the possible solutions for solving self-intersections and intersections for polygonal meshes was suggested in (Campen and Kobbelt 2010). They combined adaptive octree with nested binary space partitions for guaranteeing exactness and robustness of the intersection operations or construction of outer hulls which can be suitable for a mesh repair tasks.

Polygonal meshes provide only an approximate description of the object that is highly dependent on the amount of triangles/polygons. They are not well suited for defining heterogeneous objects as polygonal meshes define only the surface of the object, not its interior. One of the possible solutions was introduced in (Lei et al. 2014) where nested shells were suggested to use. Alternatively, in some CAD systems the interior of the polygonal objects can be defined as a homogeneous material, but it does not cover an uneven attribute distribution in interior of the object.

Summary

In this section we have reviewed one of the widely used representations - the boundary representation. This representation can be defined using parametric or polygonal surfaces. However, as in this work we are considering volumetric representations that are suitable for heterogeneous object modelling, we showed that BRep is not well-suited for defining such objects, and it had various problems, e.g. self-intersections and different inconsistencies.

According to the definition 2.1.1 of the heterogeneous object, the object consists of geometric shape and attributes that are defined on the boundary and in its interior. The core disadvantage of BRep for the heterogeneous multi-material volumetric modelling is that it is surface-based and the interior of the objects is hollow. Beside that, BRep is an approximated representation (discrete) and unsmooth.

2.3 Volumetric Representations

In this section, we review six types of commonly used volumetric representations, namely, voxel representation, implicit surfaces, function representation (FRep), signed distance fields (SDFs), adaptively sampled distance fields (ADFs) and interior distance fields (IDFs) that are suitable for heterogeneous object modelling. We identify their advantages and drawbacks. We also review the state-of-the-art methods for their generation. At the end of this section we discuss hybrid approaches for volumetric modelling presented in the literature.

2.3.1 Voxel Representation

Boundary approximating representations can be defined as grids, voxels and hierarchical grids (e.g. octrees) (Goodman, O'Rourke, and Tóth 2017). The grids can be of two types, namely regular or adaptive. Regular grids subdivide the n-dimensional Euclidean space into cubes. Voxels are rectilinear grids, where each element has equal size. They can also be defined using adaptive or hierarchical grids for efficient representation of the volumetric object. Each voxel can store a scalar value (e.g defining density of the material in the point) or a vector value (e.g defining colour, velocity vector etc). This is one of the common data structures for storing and processing volumetric data and for defining multi-material heterogeneous objects.

In Fig. 2.6 we show two examples of how voxel representation can be used. In Fig. 2.6 (a) we can see a textured BRep airplane model that was converted to voxel representation. Voxels can also be used for 3D painting. In Fig. 2.6 (b) of this figure we show a picture of the robot that was drawn in a voxel-based painter *Magicavoxel* (Simar 2020). The model in the picture is made of fixed-size voxels that store an attribute property (color, textures, etc) specified by an artist.

However, voxels approximate data, and the level of detail highly depends on the voxel grid resolution. The data stored in voxels is discrete and, generally, consume a lot of memory.

As using regular voxel grids is inefficient in terms of memory handling, more researchers turn their attention to the sparse adaptive solutions. One of them is GigaVoxels introduced in (Crassin et al. 2009). It is based on generalised octrees (N^3-trees) with MIP-mapped 3D texture bricks in its leaves. This approach can utilise both CPU and GPU computational capabilities. It handles the cases when the details are concentrated at interfaces between dense clusters and empty space.

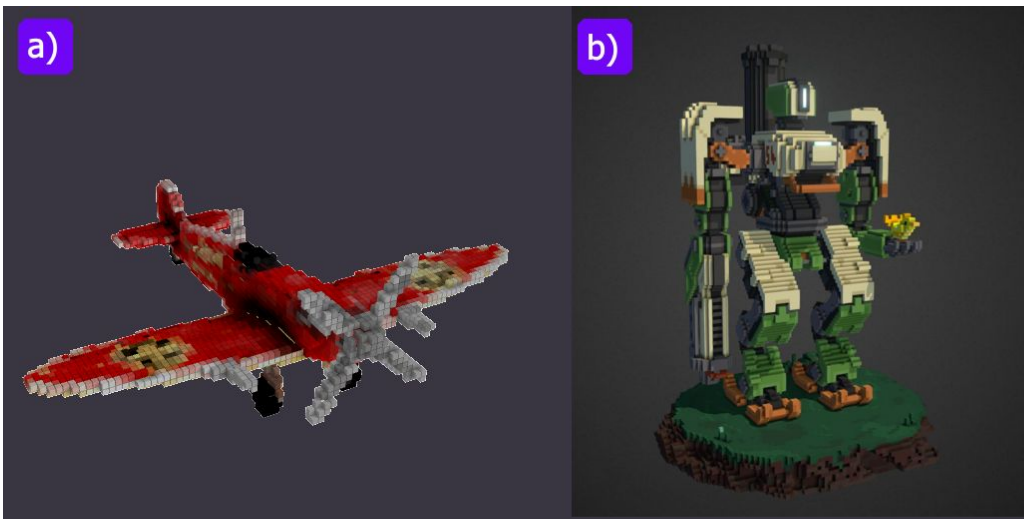


Figure 2.6: *Voxel based models. a) the voxelised model of the 'Red Baron' plane (BRep model)* [a]*; b) an example of the voxel-based art* [b]*.*

[a]The plane model (`https://sketchfab.com/3d-models/red-baron-c1352e0038d8449999e5d0cebb42786f/`) was used under CC Attribution-ShareAlike licenses `https://creativecommons.org/licenses/by/4.0/`

[b]The robot model (`https://sketchfab.com/3d-models/bastion-ganymede-overwatch-b25da6e53d304c00a6bda81d26114ac5`) was used under CC Attribution-ShareAlike licenses `https://creativecommons.org/licenses/by/4.0/`

Unfortunately, this method is not suitable for handling dynamic objects. In (Laine and Karras 2010) another octree-based solution that uses voxels was introduced. In particular, authors state that the smoothness of the voxelised surface of the object is not necessarily important. The method suffers from the discontinuities at voxel boundaries that could introduce problems such as self-shadowing or interreflections during ray tracing.

One of the most widely used voxel-based libraries presented in the industry is OpenVDB (Museth 2013) which was developed by DreamWorks Studio. It is an adaptive data structure that shares some characteristics with B+trees and facilitates both uniform and adaptive sampling. It is memory efficient, supports time-varying data, and it is suitable for representing volumetric object with its interior structure and attributes. It is fully utilising CPU for multi-threading computations. OpenVDB provides flexible and efficient control of the data, that is, unfortunately, still discrete and approximate. Another promising voxel-based library was released quite recently by NVIDIA and called GVDB (Hoetzlein 2016). This library is based on the voxel database topology introduced in (Museth 2013) and it is also adaptive. The main goal of GVDB is the full utilisation of GPU parallel computations for data processing, particularly for handling time-variant objects and rendering.

To enhance memory efficiency while working with voxel representation, the stored data can be compressed. In (Kämpe, Sintorn, and Assarsson 2013) the directed acyclic graph (DAG) based structure was introduced to significantly reduce the size of the sparse voxel octree. DAG solves the problem of the poor scaling with respect to resolution. However, it is only suitable for handling the static data that is still represented as discrete blocks. Later DAG method was extended in (Dado et al. 2016; Dolonius et al. 2019) to include compressed per-voxel simple attributes (e.g. colours). In (Careil, Billeter, and Eisemann 2020) the DAG approach was further enhanced to enable interactive modifications of the stored compressed data without

Advantages	Drawbacks
• Can be defined using regular and adaptive grids; • Can store a lot of information about the object geometry and its attributes; • Efficient to process; • Can be used for dynamic simulations, e.g. particle simulations; • Efficient to render using physically-based techniques;	• The stored data in voxels is essentially approximated (geometry and attributes) and discrete; • Voxel representation is not smooth and can be discontinuous; • Memory inefficient; • Resolution dependent: the density of the voxel grid controls the geometry quality of the defining object;

Table 2.2: *Advantages and drawbacks of the voxel representation.*

its decompression including compressed attributes.

2.3.2 Implicit Surfaces

Implicit surface S (Bloomenthal and Wyvill 1997) is defined as a set of points in Euclidean space $\mathbb{R}^n$ where a scalar field equals to the given iso-contour value $c \in \mathbb{R}$:

$$S = \{\boldsymbol{p} \in \mathbb{R}^n / f(\boldsymbol{p}) = c\} \tag{2.12}$$

where $\boldsymbol{p} = (x_0, x_1, ..., x_n)$ is a point in Euclidean space $\mathbb{R}^n$. Implicit volume (Schmidt et al. 2007) V is similarly defined:

$$V = \{\boldsymbol{p} \in \mathbb{R}^n / f(\boldsymbol{p}) \geq c\} \tag{2.13}$$

One of the important early works on implicit surfaces is Blinn's work (Blinn 1982) where they were used for modelling and rendering of electron density maps of a molecular structure. Electrons were represented by blobby objects, which were defined as the sum of Gaussian bumps. The shape of the blobby object could be changed by varying Gaussian bumps parameters. In Fig. 2.7 we show two examples of implicit surfaces. In Fig. 2.7 (a) we show an object that consists of five blobby objects and in Fig. 2.7 (b) we show five tori objects combined together.

Wyvill at al. (Wyvill, Mcpheeters, and Wyvill 1986) generalised blobby objects and introduced soft objects. He suggested to define such objects as a surface of constant value in a scalar field over three dimensions. He modified the exponential function in the defining function of the blobby object by approximating it with piecewise polynomial expressions.

Authors (Turk and O'Brien 2002) suggested to use an interpolating implicit surfaces which are smooth, satisfy the boundary constraints introduced in the interior, on the boundary, and in exterior of the object. This implicit representation defined closed surfaces of an arbitrary topology.

In (Shapiro and Tsukanov 1999) implicit surfaces defined by normalised distance-based functions with a guaranteed differential property were described. It was stated that there was a problem of interior zeroes in the scalar field generated using such functions. The possible solution for it was found in (Shapiro 1999).

The BlobTree method suggested in (Wyvill, Guy, and Galin 1999), was based on an implicit surface modelling system. In this work, the authors described an implicit modelling method that used an expendable tree data structure for handling geometry processing. However, the application of the Boolean operations to the

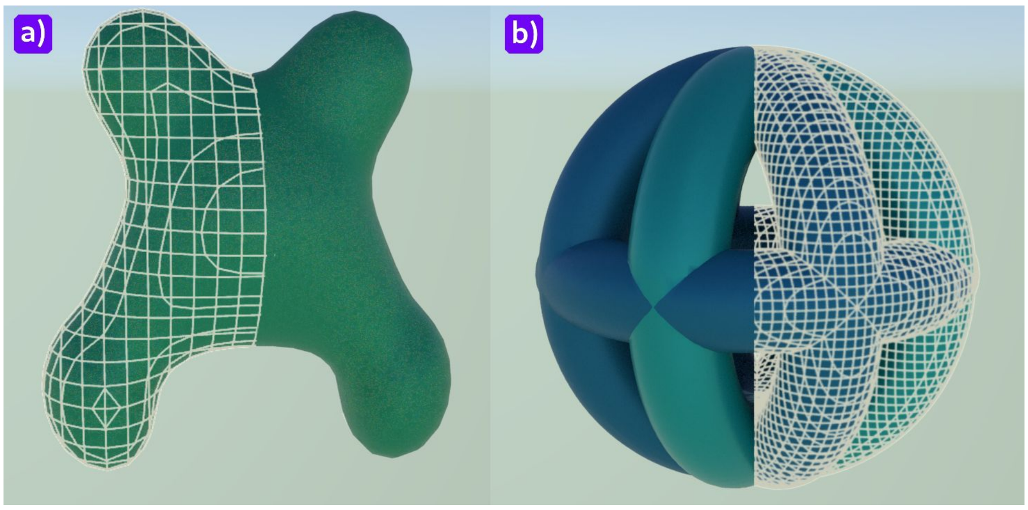


Figure 2.7: *The examples of the implicit surfaces with their polygonal grid that were generated by the author in SideFX Houdini. a) five blobby objects united in one object; b) five tori objects combined in one object.*

Advantages	Drawbacks
• At least C^0 continuous functionally defined objects; • Smooth object representation; • Suitable for defining attribute distributions in interior of the object; • Can be efficiently defined as a solution of the partial differential equation;	• Problem of interior zeroes; • Set-theoretic operations can introduce discontinuities or junctions between surface elements;

Table 2.3: *Advantages and drawbacks of the implicit surfaces.*

implicitly defined objects introduced discontinuities or junctions between surface elements (Wyvill, Guy, and Galin 1999).

An interactive shape design can be achieved using volumetric implicit partial differential equations (PDEs) (Du 2003). This method has the following advantages: by solving PDEs, both boundary and interior information can be obtained at the same time; higher-order continuity can be provided for the geometric objects; optimisation techniques can be used with the PDE models. Smooth results with tangential continuity can be obtained using elliptic PDEs. The suggested method could also handle material distributions inside the object (Du and Qin 2001).

The level-set method is another widely used technique for modelling implicit surfaces introduced by Osher and Sethian in (Osher and Sethian 1988). Level-set method is PDE based. The solution of PDE is defined as a continuous function that evolves in time. This method handles topological changes and reduces the noise in the data set. The level-set method can also be used for representing non-manifold surfaces. In (Yuan, Yu, and Wang 2012) a novel class of object-space multi-phase implicit functions was introduced. These functions were capable of accurate and compact representation of the objects with multiple internal regions.

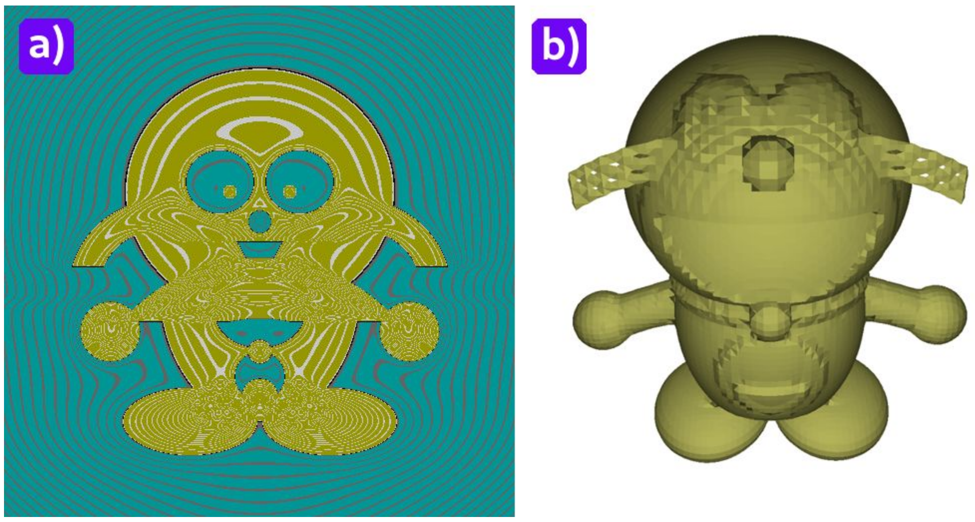


Figure 2.8: *Two objects that were defined using function representation. a) a 2D character that was defined using 42 set-theoretic operations; b) a 3D cat character that was created using HyperFun language (Adzhiev, Cartwright, et al. 1999).*

2.3.3 Function Representation

The function representation (FRep) (Pasko, Adzhiev, Sourin, et al. 1995) defines a geometric shape G of the object O_{FRep} as a closed subsets of an n-dimensional Euclidean space $\mathbb{R}^n$ using a real-valued defining function $F_{FRep}(\boldsymbol{p})$ that is at least C^0 continuous (the formal definition for FRep will be given in section 3.2.1). The higher order of continuity guarantees the existence of derivatives and, as a consequence, the smoothness of the function (this will be discussed in more details in section 3.1).

FRep provides the information about point-membership (see Fig. 2.8, a), i.e. it is positive inside, takes exact zero on the boundary of the object and negative in exterior of the object. FRep is a high-level and uniform representation of multidimensional geometric objects. The subject of particular interest is 4D objects with fourth coordinate specified as a time.

FRep generalises implicit surface modelling and extends a constructive solid geometry (CSG) approach. FRep has a closure property as operations applied to the FRep defining functions produce continuous resulting FRep functions. The FRep object can be defined as a primitive (e.g sphere, octahedron, cylinder, etc.) or as a complex object that is defined in the form of a constructive tree. In this case, primitives are stored in the leaves of the tree and operations are stored in its nodes.

There exist many well-developed operations, e.g set-theoretic operations, metamorphosis, blending and bounded blending and space-time blending (Pasko, Pasko, and Kunii 2005), offsetting, bijective mapping and others (Pasko and Adzhiev 2004). FRep covers traditional solids (Karczmarczuk 1999), scalar fields, heterogeneous objects including both static and time dependent volumes (Ďurikovič, Czanner, and Inoue 2001). FRep is also suitable for defining volumetric microstructures (Pasko, Fryazinov, et al. 2011) as well as for defining multi-scale cellular structures (Fryazinov, Vilbrandt, and Pasko 2013). In (Pasko, Adzhiev, Schmitt, et al. 2001) FRep was also extended for heterogeneous volumetric modelling using hypervolume representation. The FRep function can be defined analytically, or with function evaluated algorithm, or with a discrete scalar field. The defining function can be generated, for example, for point clouds with an interpolating surface (Fayolle and Pasko 2016) and mesh surfaces (Fryazinov, Pasko, and Adzhiev 2011). FRep can be extended

Advantages	Drawbacks
• FRep generalises implicit surface modelling and extends a constructive modelling approach; • FRep supports point membership; • FRep is closed guaranteeing to get at least C^0 continuous resulting function; • FRep covers solids, scalar fields, volumes, time-dependent volumes and hypervolumes for heterogeneous object modelling. • FRep has many well-developed operations that support multidimensional transformations in $\mathbb{R}^n$;	• Distances can be obtained for a limited number of FRep objects; • FRep object can have a boundary with dangling portions that are not adjacent to the interior of the object; • FRep has an unpredictable non-distance based behaviour of the resulting field and, as a consequence, it is problematic to render in 3D.

Table 2.4: *Advantages and drawbacks of the function representation.*

with several other representations such as voxels, meshes and point clouds that can be stored as primitives in the leaves of the FRep constructive tree (Adzhiev, Kazakov, et al. 2000). This makes FRep essentially a hybrid representation of volumetric objects as not only an object surface, but its interior is defined by a function.

In Fig. 2.8 we show 2D and 3D objects that were defined using FRep functions and set-theoretic operations. In Fig. 2.8 (a) we show how the FRep field varies in exterior and interior of the object using a 2D example. In general case, the FRep field is not distance-based as field isolines do not precisely follow the object shape. In Fig. 2.8 (b) we show a 3D FRep object that was generated using *HyperFun* tools (Adzhiev, Cartwright, et al. 1999). The advantages and drawbacks of the representation can be found in table 2.4.

2.3.4 Radial Basis Functions

Radial basis functions (RBFs) are widely used in the context of the approximation of the multivariate functions. For instance, RBFs are used in the case when a multidimensional data defined in points $\boldsymbol{p} \in \mathbb{R}^n$ using some function $f_p = f(\boldsymbol{p})$ is given, it is essential to find an approximate function $f_s : \mathbb{R}^n \mapsto \mathbb{R}$ to the function $f_p : \mathbb{R}^n \mapsto \mathbb{R}$ from which the data is assumed to stem (Buhmann 2003). Formally, RBFs are defined as follows:

Definition 2.3.1. *(Radial basis functions) A radial basis function is a real-valued function $\varphi(\boldsymbol{p})$ that depends on the distance between the input and either the origin $\varphi(\boldsymbol{p}) = \varphi(||\boldsymbol{p}||)$ or some other fixed point $\boldsymbol{p}_c$ called a centre, i.e. $\varphi(\boldsymbol{p}) = \varphi(||\boldsymbol{p} - \boldsymbol{p}_c||)$. The sum of such radial functions forms a basis for some function space to approximate the input data represented by function values $f_s(\boldsymbol{p})$ as follows:*

$$f_s(\boldsymbol{p}) = \sum_{j=1}^{m} \lambda_j \varphi(||\boldsymbol{p} - \boldsymbol{p}_j||), \quad \boldsymbol{p} \in \mathbb{R}^n \tag{2.14}$$

where λ_j are scalar parameters, $||\cdot||$ denotes a norm (e.g. Euclidean).

RBF functions are widely used in the context of the data approximation defined on unstructured grids, e.g. the reconstruction of implicit surfaces out of the 3D scanned point clouds. In (Carr et al. 2001) polyharmonic RBFs were used for the reconstruction of the smooth, manifold surfaces from point-cloud data with repairing holes in the reconstructed mesh. The proposed greedy algorithm reduces the

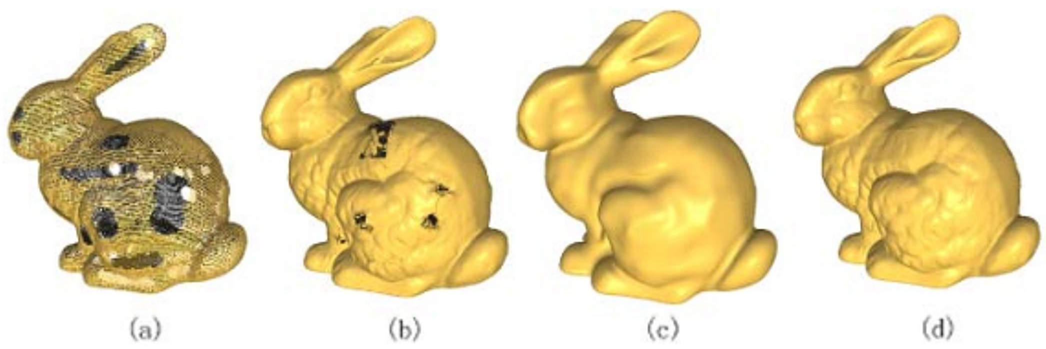


Figure 2.9: *A 'Stanford Bunny' model that has holes (a) which were closed as the result of applicaiton of the single level Duak-RBF operation (b), of the coarse level Dual-RBF operation (c), of the multi-level Dual-RBF operation (d). Reprinted by permission from Springer Nature: The Visual Computer (Lin et al. 2009) ©2009.*

amount of required RBF centres, and the energy minimisation characterisation of polyharmonic splines produces a smooth approximation of the surface. In another work (Li, Wills, et al. 2004) was introduced an algorithm for fitting implicit surfaces to a given data set using RBFs with ellipsoid constraints. The fitted shape is able to capture the main features of the object in case, when the data sets are extremely sparse. Authors (Pan and Skala 2007) described a method for reconstructing of the implicit function out of the point cloud with associated surface normals with point membership information, i.e. whether point belongs to the interior or exterior of the reconstructed object. In (Samozino et al. 2006) an algorithm for the surface reconstruction using Voronoi centred radial basis functions was proposed. The centres of RBF functions were selected among the vertices of the Voronoi diagram of the sample data points. Authors (Lin et al. 2009) introduced a method for surface reconstruction from the input point cloud using dual-RBFs. The idea of the method was based on the physical polar field model that was combined with a multi-level surface reconstruction strategy to effectively address the problem of holes filling. The result of appliacation of this method to the model of the 'Stanford Bunny' with holes can be seen in Fig. 2.9.

In general case, RBF functions tend to oversmooth data, they might be not efficient to compute. However, this type of functions provides a smooth and continuous approximation of the reconstructed surface or function. RBFs can be used for approximating functions defined on unstructured grids. In the context of this thesis, they can be used as an intermediate step for obtaining a FRep or SDF function from the input point cloud.

2.3.5 Signed Distance Fields

The signed (oriented) distance function (SDF) $F_{SDF}(\boldsymbol{p})$ of a set X defines a distance from the given point $\boldsymbol{p} \in X$ to the boundary ∂G of the geometric set $G \subseteq X$, where the sign determines whether the point $\boldsymbol{p}$ belongs to $X \backslash G, \partial G$ or interior G_{in} of the object. A more formal definition of the SDF representation will be given in subsection 3.2.2. In this work we assume that SDF point membership rule is in compliance with the FRep point membership rule.

The SDF function is at least C^0 continuous as it might not be differentiable at some points of Euclidean space $\mathbb{R}^n$ and it might has gradient discontinuities

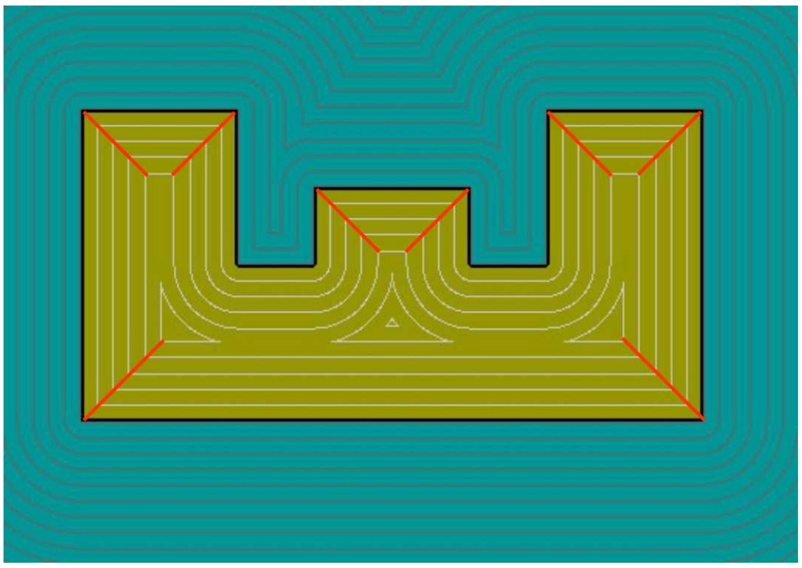


Figure 2.10: *An example of the 2D shape that has gradient discontinuities on the medial axes that are marked with red lines.*

on the object's medial axes. For instance, in Fig. 2.10 we mark this type of C^1 discontinuities using red lines. As the SDF function satisfies the solution of the eikonal equation $||\nabla F_{SDF}|| = 1$, its gradient becomes discontinuous in points that are marked with red lines.

Formally, the SDF function is Lipschitz continuous and Frećhete differentiable that were proven, for example, in (Delfour and Zolsio 2010). Lipschitz continuity is a strong implication of the function continuity that guarantees that the function is uniformly continuous. Frećhete differentiability is a generalised directional derivative of the vector function and the SDF function is a vector or a oriented function. A formal definition for both notions will be given in section 3.1. Both properties prove that SDF function is differentiable everywhere except those points where the cut locus touches the surface and where the problem of gradient discontinuities arises.

There are a lot of operations defined for SDFs. One of them is offsetting (Bálint, Valasek, and Gergó 2019). A lot of operations for SDFs were defined in (Payne and Toga 1992). These are surface interpolation, multiple-object averaging, spatially-weighted interpolation, texturing, blending, set-theoretic operation and metamorphosis. Morphological operations such as erosion and dilation (Serra 1988) can be also applied to SDFs. SDF can be used for material definition in heterogeneous objects (Biswas, Shapiro, and Tsukanov 2004), additive manufacturing (Barclay, Dhokia, and Nassehi 2016), in collision detection problems, particle simulations (Kim, Kim, and Lee 2015) and others. In Fig. 2.11 we show a 2D shape and a 3D BRep object that were converted to the SDF representation. In Fig. 2.11 (a) we show the SDF field generated for the FRep defined the 2D character using distance transform algorithm. As it can be seen, the isolines are spaced equidistantly and follow the shape of the object. In Fig. 2.11 (b) we show the 3D cartoon character that was initially defined using BRep and then converted to the SDF representation using the level-set method (Museth 2013). The advantages and drawbacks of SDF can be seen in table 2.5.

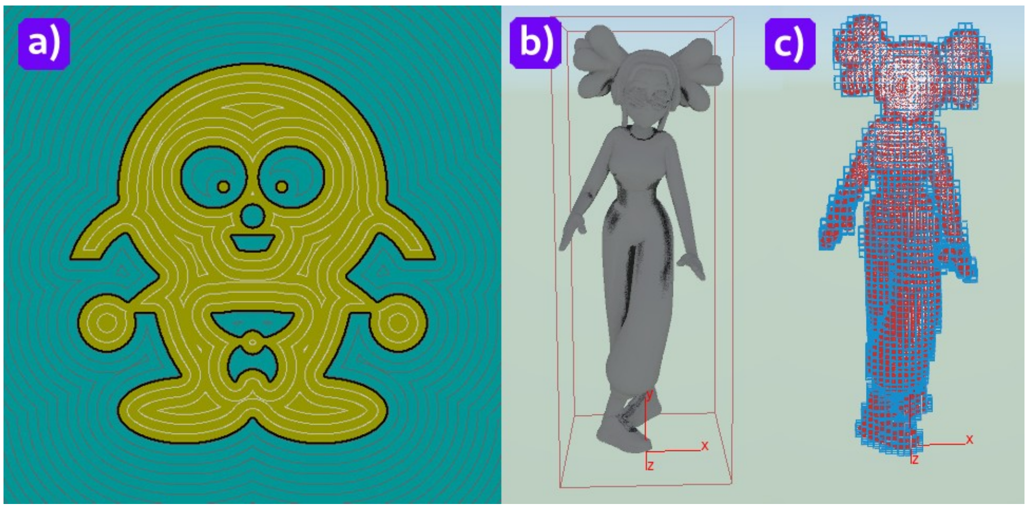


Figure 2.11: *Two objects that were converted to signed distance fields. a) a FRep 2D character that was converted to the SDF object; b) a BRep 3D cartoon model* [a] *that was converted to SDF level-set representation using OpenVDB library in SideFX Houdini.*

[a]This a free to use model https://www.mixamo.com/#/?limit=96&page=1&type=Character.

Advantages	Drawbacks
• SDF provides distances to the object surface both inside and outside it; • SDF is at least C^0 continuous; • SDF is a Lipschitz continuous function; • SDF is Frećhet differentiable almost everywhere; • SDF satisfies the solution of the eikonal equation; • SDF supports point membership; • SDF is effectively discretised, has a predictable field behaviour and is efficiently rendered.	• SDF might not be differentiable at some points of Euclidean $\mathbb{R}^n$ space. Loss of SDF differentiability happens when the current point is sufficiently close to a concave singularity (a concave corner/edge); • SDF has discontinuous gradients on the object's medial axis; • SDF is not smooth; • SDF might not be suitable for attribute modelling due to C^1 discontinuity.

Table 2.5: *Advantages and drawbacks of the signed distance function representation.*

Approximate Methods for Computing SDFs

The most simple way for obtaining a discrete unsigned or signed distance field in 2D/3D Euclidean space is to calculate the distance transform (DT). This technique provides a mapping from one scalar field $f_1(\boldsymbol{p}_1)$ to another scalar field $f_2(\boldsymbol{p}_2)$ such that $f_2(\boldsymbol{p}_2) = \inf_{\boldsymbol{p}_1 : f_1(\boldsymbol{p}_1=0)} |\boldsymbol{p}_1 - \boldsymbol{p}_2|$, where $\boldsymbol{p}_1 = (x_1, x_2, ..., x_n)$ and $\boldsymbol{p}_2 = (x_1, x_2, ..., x_n)$ are points in an n-dimensional Euclidean space $\mathbb{R}^n$. The distance obtained by DT is usually unsigned.

The DT technique was first introduced in (Rosenfeld and Pfaltz 1966) by Rosenfeld and Pfalz. They suggested to approximate Euclidean distance using a local distance propagation which considerably reduces the execution time with a trade-off in accuracy. Later this method was enhanced in several works (Borgefors 1984, 1986, 1996; Svensson and Borgefors 2002). The main idea of approaches was to use the weighted optimal distances and proper sweeping templates (see Fig. 2.12)

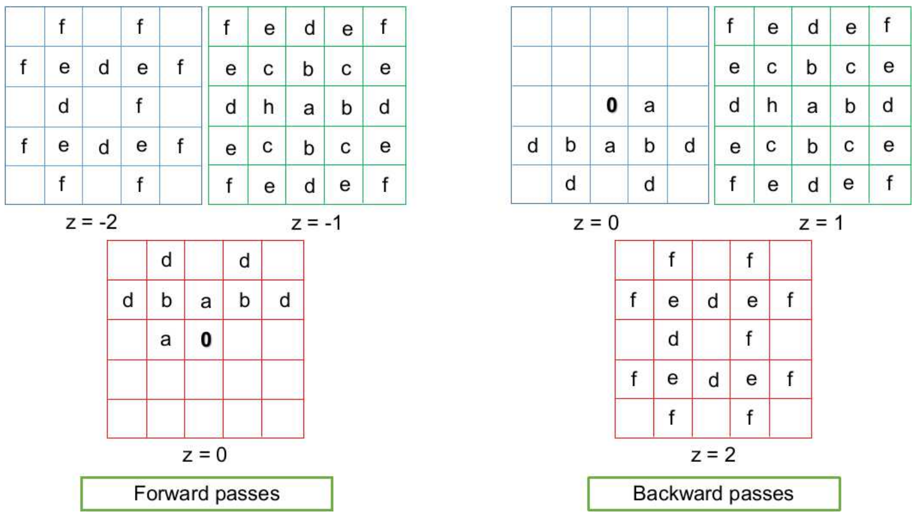


Figure 2.12: *A chamfer distance template. In the forward pass distances a-f are added to voxels in the current, $z-1$ and $z-2$ slices. In the backward pass, distances are added to voxels in the current, $z+1$ and $z+2$ slices (Jones, Baerentzen, and Sramek 2006), copyright line ©2006 IEEE.*

to obtain the resulting DT and minimise the difference between Euclidean distance and the obtained optimal weighted distance.

A more sophisticated and accurate way of calculating DT is to use the vector distance transform (VDT). A particular case of VDT is the sequential Euclidean distance transform (SEDT) introduced in (Danielsson 1980) for 2D images. He suggested two versions of the algorithm that use four or eight neighbour points defined in the 3×3 template. Four sweeps of the template were applied to the computational grid. During each sweep the computed distance in the current grid point was compared with distances stored in the neighbour points. Then the minimised Euclidean distance was stored in the current grid point. The SEDT algorithm was improved in two works (Leymarie and Levine 1992; Ye 1988) where the signed vectors for calculating DT were introduced. In (Mullikin 1992) the four-point signed SEDT algorithm into three dimensions was extended. The obtained DT algorithm was called the efficient VDT algorithm (EVDT). It used six sweeps of the defined template matrix and can be applied to an anisotropically sampled data. Later the EVDT algorithm was enhanced in (Satherley and Jones 2001) where the unsigned vector-city VDT (VCVDT) algorithm was introduced. Authors suggested an eight passes algorithm (four forward passes and four backward passes) with a complete vector copy of the distance field.

SDFs can be computed using a number of different approximation techniques. In (Biswas and Shapiro 2004) an approximate distance fields with non-vanishing gradients were introduced. The main idea of this work was to use the normalised approximation of the distance fields using piece-wise polynomials and to normalise distance fields using the method introduced in (Rvachev 1982). Unfortunately, it still might produce gradient discontinuities and consumes plenty of memory.

The signed approximate real distance functions (SARDF) were introduced in (Fayolle, Pasko, and Schmitt 2008). In this work, the SDF modelling was extended with set-theoretic operations preserving a distance property and being smooth at the same time. In (Belyaevm, Fayolle, and Pasko 2013) signed L_p-distance fields for SDF approximation using a family of generalised double-layer potentials were introduced. Unfortunately, this approach has relatively high approximation errors near the sharp features of the object. In (Sanchez, Fryazinov, Fayolle, et al. 2015) SDFs were approximated using a convolution filtering method. The generated SDF function is smooth and at least C^1 continuous.

Numerical Methods for Computing SDFs

The SDF field can be computed using numerical solutions of PDEs. One of the equations that can be used for the numerical computation of SDF is the eikonal equation. The eikonal equation is a non-linear PDE that is used for solving the problems of wave propagation. Formally, it is defined as follows:

$$|\nabla f(\boldsymbol{p})| \cdot f_{sp}(\boldsymbol{p}) = 1 \tag{2.15}$$

where $f_{sp}(\boldsymbol{p})$ is a speed function, $f(\boldsymbol{p})$ is the shortest time that is needed to travel from the boundary of the set to its point $\boldsymbol{p}$. In general case, speed function $f_{sp}(\boldsymbol{p})$ controls the motion of the wave propagation from the seed region. However, if the solution of the eikonal equation is computed in the context of the geometric modelling, then $f_{sp}(\boldsymbol{p}) = 1$. In this case the solution of this equation gives the signed distance from the boundary defined by $f(\boldsymbol{p}) = F_{SDF}(\boldsymbol{p})$.

One of the important properties of the SDF representation is that its defining function satisfies the eikonal equation (see table 2.5). This property shows that the Euclidean distance is preserved. From this equation (2.15) also follows that the SDF function has discontinuous derivatives in the points where its gradient is discontinuous.

Another PDE that can be numerically solved to generate SDF is the level-set equation (Osher and Sethian 1988):

$$\frac{\partial f(\boldsymbol{p})}{\partial t} + \nu_n |\nabla f(\boldsymbol{p})| = 0; \quad \nu_n = \boldsymbol{\nu} \cdot \boldsymbol{n} \tag{2.16}$$

where $\boldsymbol{n} = \nabla f(\boldsymbol{p}) / |\nabla f(\boldsymbol{p})|$ is the normal to the interface (e.g. boundary of the object), $\boldsymbol{\nu} = (u, v, w)$ is a velocity field and here $f(\boldsymbol{p})$ is a level-set function. In general case, the level-set function is non-distance based as it evolves in time through equation (2.16). In (Sussman, Smereka, and Osher 1994) a reinitialisation equation to restore the distance property of the computed solution of the level-set equation was introduced. This reinitiliasation equation can be written as follows:

$$f_\tau(\boldsymbol{p}) + \operatorname{sgn}(f(\boldsymbol{p})^0)(|\nabla f(\boldsymbol{p})| - 1) = 0 \tag{2.17}$$

where $\operatorname{sgn}(\cdot)$ is a smoothed-out signum function, $f(\boldsymbol{p})^0 : \mathbb{R}^n \mapsto \mathbb{R}$ is a level-set function that is transformed into SDF $f(\boldsymbol{p})$ and τ defines a fictitious time that controls the width of the band around the zero-level set, where a signed distance will be computed.

The eikonal equation can be numerically solved using fast methods such as fast marching method, fast sweeping method, fast iterative method and others. The fast marching method (FMM) was introduced in (Tsitsiklis 1995). This method gives a solution of the isotropic control problems using a first-order semi-Lagrangian discretisation on the Cartesian grid.

Another popular method for solving the eikonal equation was introduced in (Zhao 2004) and called the fast sweeping method (FSM). It is an iterative algorithm that uses Gauss-Seidel iterations with alternating sweeping ordering to obtain the numerical solution of equation (2.15) on a rectangular grid.

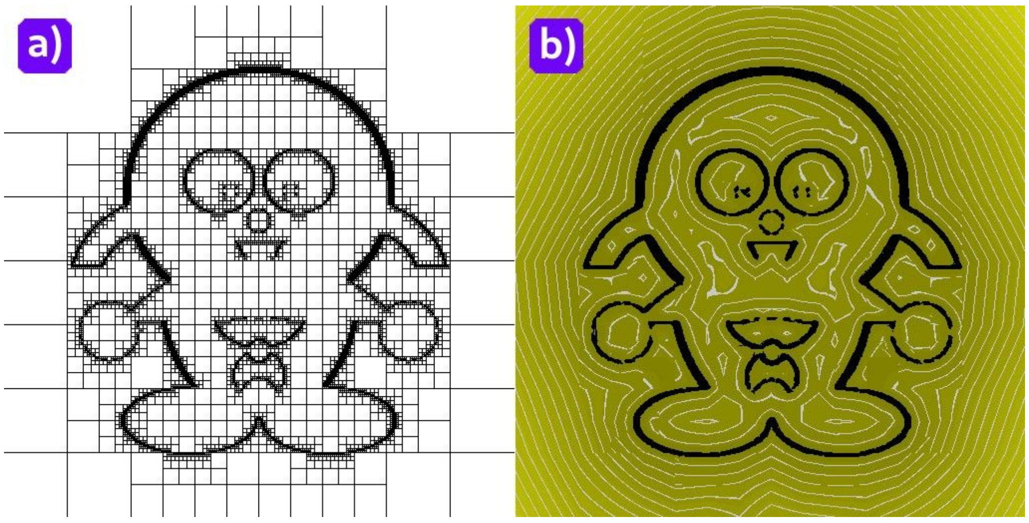


Figure 2.13: *Adaptively sampled distance field that was restored using bilinear interpolation. a) the subdivided 2D character using octree that was defined using FRep; b) the computed and restored distance field for the 2D FRep defined character.*

In (Jeong and Whitaker 2008) a fast method called the fast iterative method (FIM) was introduced. It uses a first order upwind Godunov discretisation scheme and independent update of the solution of the eikonal equation computed on a regular grid. Unfortunately, this scheme introduces bounded errors as a trade off for computational efficiency. In (Hong and Jeong 2017) an enhanced version of the FIM algorithm called group-ordered FIM (GO-FIM) was proposed. It significantly reduces redundant computations, which was the main drawback of the original FIM algorithm.

The level-set method is an implicit representation of a moving front that was introduced in (Osher and Sethian 1988). It is an efficient way of representing and tracking the evolution of the interfaces according to equation (2.16). The main advantage of this method is that it could handle various topological changes of the object. To solve the defining level-set equations (2.16) - (2.17) a total variation diminishing (TVD) Runge-Kutta scheme (Shu and Osher 1989) is used along with a Godunov spatial discretisation of Hamiltonian.

Recently, an alternative and a more efficient way to compute SDF was introduced in (Darbon and Osher 2016; Lee, Darbon, et al. 2017) using the Hopf-Lax formula. The main advantage of this method is an independent parallel computation at each grid point. In (Royston et al. 2018) this approach was used in the context of the level-set method when the interface is defined on a grid.

2.3.6 Adaptively Sampled Distance Fields

An adaptively sampled distance function (ADF) (Frisken et al. 2000) is a distance function that is computed on hierarchical grids, e.g. tree-like data structures. To our knowledge, there is no well-established formal definition of ADF in the literature. There are several works where ADF is interpreted in a different way compared to (Frisken et al. 2000). For example, in (Wang et al. 2011) ADF was defined using T-meshes with different interpolation operation for restoring the field, in (Koschier et al. 2017) it was suggested to use a hierarchical hp-adaptation for constructing ADF, in (Tang and Feng 2018) it was proposed to construct ADF using estimation of the principal curvatures of the input surface. In this work we introduce a formal

Advantages	Drawbacks
• ADF data structure efficiently subdivides the Euclidean space $\mathbb{R}^n$ according to the level of detail; • ADF distances are adaptively sampled; • ADF supports point membership; • ADF possesses an efficient memory management: in a small amount of memory a significant amount of information about the object can be stored; • ADF hierarchical tree data structure is fast to rebuild that make it possible to handle time-variant objects; • ADF can be efficiently rendered in real time.	• ADF field has C^0 discontinuities where cells of the different size appear as the result of the hierarchical subdivision; • ADF field has C^1 discontinuities that are introduced by the bilinear/trilinear interpolation (Frisken et al. 2000) during reconstruction of the field at each cell; • ADF is not suitable for attribute modelling due to C^0 and C^1 discontinuities.

Table 2.6: *Advantages and drawbacks of the adaptively sampled distance function representation.*

definition of ADF, which will be presented in subsection 3.2.3. Here we give a more informal definition that follows (Frisken et al. 2000).

The construction of the ADF function starts from subdividing the subset X of Euclidean space $\mathbb{R}^n$ according to the local details of the geometric shape $G \subseteq X$ using some k-ary tree. Each node of this tree is represented by an n-dimensional cell with vertices. We have to compute the distances at each vertex of each cell to the subdivided boundary of the geometric shape ∂G of the object. The boundary of the geometric shape is subdivided with some maximum tree depth. Then the distances are restored inside each cell using some interpolation technique.

The ADF field, generated as it was described in (Frisken et al. 2000), has C^0 discontinuities where the cells of different size appear, and it has C^1 discontinuities caused by the bilinear/trilinear interpolation that was used for restoring DF at each cell. The ADF representation also provides the information about point membership.

In Fig. 2.13 we show the ADF field that was generated for the FRep defined 2D character. In Fig. 2.13 (a) we show a subdivided 2D FRep object using a quadtree data structure and in Fig. 2.13 (b) we show a computed and restored distance field for this FRep object. The distances at each cell were restored using bilinear interpolation. In the places, where the isolines are broken, we can see C^0 discontinuities in the field and, where the field has sharp corners, we can identify C^1 discontinuities.

ADF can be used for an efficient interactive real-time modelling, e.g sculpting as the tree data structure provides fast access to object's geometry and its specified attributes (Perry and Frisken 2001). ADF is suitable for solving surface restoration problems (Deng et al. 2008; Tang and Feng 2018). It supports the same operations as SDF. ADFs can be used in dynamic simulations (Koschier et al. 2017), for example, morphing between shapes, as a hierarchical data structure can be efficiently rebuilt at each animation frame (Frisken et al. 2000). The advantages and drawbacks of ADF can be seen in table 2.6.

Methods for Computing ADFs

ADFs can be efficiently constructed using distribute computations. In (Yin, Liu, and Wu 2011) a special distance adaptive sampling procedure through the octree-

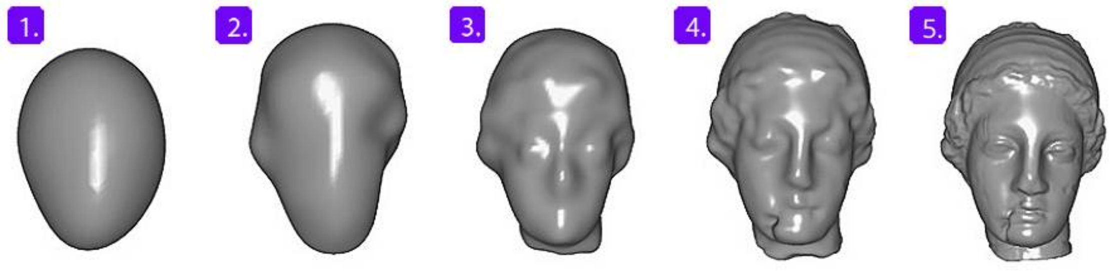


Figure 2.14: *An adaptive reconstruction of 'Igea' model using ADFs that are restored with PHT-splines at each cell of the octree. Reprinted from (Wang et al. 2011), Copyright ©2011, with permission from Elsevier.*

like structure was suggested to use. The authors introduced three sample types depending on the distance from the mesh. This approach guaranteed an efficient and accurate geometry processing. However, there were still some C^0 discontinuities at the border of the object.

Another way of setting up ADF was suggested in (Koschier et al. 2017) using the hp-refinement technique, based on piecewise polynomial fitting. The main idea of an hp-refinement is to refine the cell-size (h) while traversing the octree. Then applying the adaptive sampling step at each cell and, finally, obtain the approximation of the polynomial degree (p).

There are several works that suggested methods for solving C^0 and C^1 discontinuities in the ADF field. In (Bastos and Celes 2008) the ADF implementation with work around C^1 discontinuities was introduced. A smooth reconstruction of the surface normals is done by calculating and storing them separately from the generated object. In (Wang et al. 2011) an ADF-like approach was suggested. The distances inside each cell are calculated using PHT-splines, thus making the distance field C^1 continuous. In Fig. 2.14 we show the intermediate results of the adaptive reconstruction of the 'Igea' model using PHT-splines. As it follows from the picture the resulting surface is smooth and continuous. In (Tang and Feng 2018) ADFs were used for a multi-scale surface reconstruction. The C^1 continuity of the restored field is achieved using multi-scale B-spline functions.

ADFs can also be generated as a numerical solution of the eikonal or level-set equation on adaptive Cartesian grids. In (Strain 2000) a fast numerical node-based second-order semi-Lagrangian method for solving general moving interface problems on hierarchical grids was proposed. The author solved the advection equation (2.16) and applied the redistancing operation to the result to restore the signed distances. In another work (Min and Gibou 2007) a second-order accurate semi-Lagrangian level-set method on the non-graded adaptive Cartesian grids was proposed. The described method produces a continuous resulting distance function and negligible mass loss. In (Mirzadeh et al. 2016) a scalable algorithm for the level-set method on dynamic adaptive quadtree/octree Cartesian grids was presented. The proposed computational scheme was paralleled and a semi-Lagrangian method was applied as it is free of any time-step restrictions. A possible alternative data structure to quadtree/octree is a hash table (Brun, Guittet, and Gibou 2012) that only stores a band of grid points adjacent to the interface. Instead of tree structure, in (Guittet et al. 2015) the Voronoi interface method was used for subdividing the space. The numerical method, suggested in that paper, was applied for solving elliptic problems with discontinuities while the solution remains continuous and smooth.

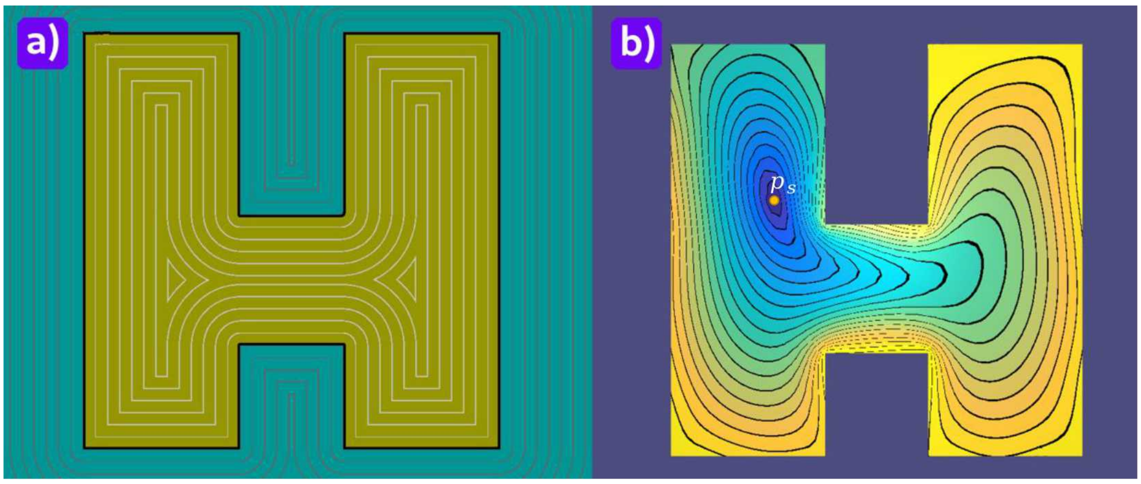


Figure 2.15: *The SDF (a) and IDF (b) fields computed for the 2D polygon. This example was generated by the author using method described in (Rustamov, Lipman, and Funkhouser 2009).*

There is lack of fast methods that were used for the computation of the eikonal equation solution on hierarchical grids. Most of the methods were developed either for triangulated meshes (Fu et al. 2011) (FIM) or tetrahedral meshes (Lelièvre, Farquharson, and Hurich 2011) (FMM). However, a simplex free adaptive FSM method for solving the level-set equation on n-dimensional binary tree data structure was introduced in (Cecil, Osher, and Qian 2006).

2.3.7 Interior Distance Fields

Interior distance field (IDF) is not a well-established notion yet as in literature there is neither a general approach for generating distance functions of this rather broad nature nor one unique name for them. In this work we suggest to use this notion for a representation with a defining function obtained as follows: the distance function is computed on the boundary of the object as a shortest path between boundary points and then the generated distances are smoothly interpolated in the object's interior. The formal definition will be given in the next chapter, subsection 3.2.4.

IDF is usually obtained by solving a partial differential equation (PDE) or applying some numerical method, e.g graph approaches (Liu, Ramani, and Liu 2011) or Markov chains (Coifman and Lafon 2006). Among PDE-based methods the following methods can be considered as representative: geodesic distances obtained as the solution of heat equation (Crane, Weischedel, and Wardetzky 2013), diffusion maps combined with smooth barycentric interpolation of the distances in interior of the object (Rustamov, Lipman, and Funkhouser 2009), the optimal mass transport (Solomon, Rustamov, et al. 2014) and some others. IDF is usually used in the tasks related to shape analysis (Rustamov, Lipman, and Funkhouser 2009), geometry restoration (Patanè and Spagnuolo 2012), morphing and less commonly for an attribute definition in interior of the object (Fryazinov, Sanchez, and Pasko 2015). IDF advantages and drawbacks can be seen in table 2.7.

SDFs and IDFs have very different distance field isolines due to the methods that are used for their generation. In case of SDF, we compute the distances to the surface of the object, while in case of IDF, the distances are computed on the boundary and then extended to interior of the object.

In Fig. 2.15 we show a 2D comparative example. In Fig. 2.15 (a) we show an

Advantages	Drawbacks
• IDF is at least C^0 continuous; • IDF is shape-aware; • IDF is deformed with the boundary; • IDF is smooth; • IDF is suitable for distance-based attribute definition in interior of the object.	• IDF can be computationally expensive; • IDF field accuracy for some methods is highly dependent on a time step and type of the used discretisation; • IDF is defined only in interior of the object.

Table 2.7: *Advantages and drawbacks of the interior distance function representation.*

SDF field obtained for the 'H' object. We can see that it is defined in interior as well in exterior of the object. It has C^1 discontinuities where the sharp features can be seen in interior of the object. On the contrary, in Fig. 2.15 (b) an IDF field is computed between 'source' point $\boldsymbol{p}_s$ and the other points of the 2D shape and is defined only in interior of the shape and on its boundary. As it can be seen it is smooth, shape-aware and completely different comparing to SDF. In Fig. 2.16 we show a 3D comparative example of SDF and IDF fields that were computed for the 'Stanford Bunny' model. In Fig. 2.16 (a) we show the IDF field computed on the boundary of the object, and in (b) and (c) we show two mesh slices with computed IDF and SDF fields in its interior. The IDF field is smoothly varies in interior of the object while SDF does not.

Methods for Computing IDFs

In work (Rustamov, Lipman, and Funkhouser 2009) the generation of IDF is based on the propagation of the distances computed on the boundary of the mesh in its interior. The boundary surface of the mesh was embedded in a higher dimensional space using diffusion maps (Coifman and Lafon 2006). The diffusion maps approximate Euclidean distance on the boundary. Then the computed boundary distances are propagated in the interior of the mesh. This method is quite resourceful for computations but provides accurate results.

In (Crane, Weischedel, and Wardetzky 2013) IDF was obtained using the solution of the heat equation. The suggested method consists of three steps: integrating the heat flow for some fixed time, then calculating the vector field and, finally, solving the Poisson equation. The accuracy of this method depends on the particular choice of the spatial discretisation. In work (Rustamov 2011), the interior heat kernel signature was obtained using Laplace-Beltrami eigenfunctions and barycentric coordinates. The interior distances were obtained by propagating Laplace-Beltrami eigenfunctions defined on the boundary of the mesh, inside the mesh, using barycentric interpolation. This representation provides a continuous distance field, and it is insensitive to deformations.

Yu-Shen Liu and others (Liu, Ramani, and Liu 2011) suggested a graph-based method for generating IDFs. The inner distance is calculated as the length of the shortest path of the graph between landmark points. The obtained distances are invariant to articulations and sufficiently accurately approximate Euclidean distance. Unfortunately, these distances are very sensitive to concavity/convexity changes of the shape. Authors (Fryazinov, Sanchez, and Pasko 2015) also used a graph-based

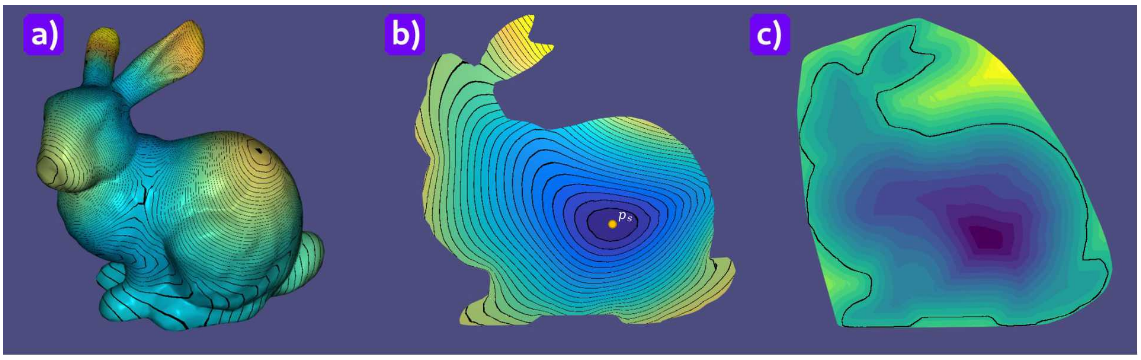


Figure 2.16: *The IDF field computed for the 'Stanford Bunny' 3D tetrahedralised mesh using the method described in (Rustamov, Lipman, and Funkhouser 2009) and an SDF slice to show the difference in nature of these fields. a) the IDF field computed on the boundary of the mesh. Black isolines show how the field is changing according to the shape of the object; b) the interior slice of the mesh with computed IDFs. The yellow point $\boldsymbol{p}_s$ in the slice corresponds to the 'source' point. c) the SDF slice of the same model with computed interior and exterior distances. Colour changing reflects how the distances are changing from interior to exterior of the object.*

algorithm for calculating IDFs and setting up materials in the interior of the object. The distances were computed as an approximate Euclidean shortest path between a point and the material feature. The obtained distance field was continuous and differentiable everywhere.

A new method for calculating Earth Mover's Distances was introduced in (Rubner, Tomasi, and Guibas 2000) and enhanced in (Solomon, Rustamov, et al. 2014). It can be used for the IDF calculation. They applied the theory of optimal mass transportation and passed dual differential formulation with linear scaling. The resulting distances are smooth, continuous and can be efficiently computed.

IDFs can also be obtained using spectral distances (Patané 2016; Patanè and Spagnuolo 2012). One of the main properties of the spectral distances is their capability of storing local geometric properties of the input shape, shape-awareness, robustness to noise and tessellation.

2.3.8 Hybrid Representations

The main feature of any hybrid representation is that it unifies advantages of several representations and compensate for their drawbacks. On the basis of FRep several hybrid approaches were introduced. One of them is hybrid volumes (Adzhiev, Kazakov, et al. 2000) which were defined as a combination of FRep and voxel representations. In (Pasko, Adzhiev, Schmitt, et al. 2001) the concept of hypervolumes was introduced. It was an extension of the general object model (Kumar, Burns, et al. 1999) and unified the advantages of FRep and hybrid volumes. A heterogeneous object was defined as an n-dimensional point-set with specified attributes, operations and relations over them.

In (Adzhiev, Kartasheva, et al. 2002) a hybrid cellular-function representation was proposed. It was based on hypervolumes combined with the cellular representation and the constructive representation using FRep. The concept of implicit complexes was introduced in (Kartasheva et al. 2008) which represented an object with different type of cells that defined not only geometry of the object, but also its attributes. This model makes it possible to represent dimensionally non-homogeneous elements and their cellular representations. The authors showed that attributes may define not only the material, but any volumetric distribution such as density or temperature. Another hybrid approach called hybrid surface representation was introduced in (Kim, Sukhatme, and Desbrun 2004). It was based on BRep and an implicit surface representation. This representational scheme was used for heterogeneous volumetric modelling and sculpting. Using image-based haptic techniques with introduced hybrid approach the user is provided with tool to paint on the surface of the object.

There is a number of works that describe hybrid representations based on SDFs, ADFs and IDFs. The authors (Tsukanov and Posireddy 2011) introduced a hybridisation of meshfree, RBFs, DFs and collocating technique for solving engineering analysis problems. The proposed technique enables exact treatment of all boundary conditions and can be used with both structured and unstructured grids. Sullivan (Sullivan 2015) introduced hybrid ADFs which represented the object as a set of cells.

The authors (Allegre et al. 2004) defined a new structure called HybridTree that was based on an extended CSG tree. The HybridTree structure unified advantages of skeletal implicit surfaces and polygonal meshes. The hybrid biharmonic distances that were defined similarly to diffusion and commute-time (graph) distances were proposed in (Lipman, Rustamov, and Funkhouser 2010) for solving some shape analysis tasks. This type of distances provides both reasonable global and local properties of the shape. They are smooth, but very expensive to compute. In (Kim, Kim, and Lee 2015), a concept of the hybrid ADF was introduced for detailed representation of the dynamically changing liquid-solid mixed surfaces. Solid and liquid parts of the object were combined using their level-set values.

Summary

In this section we have discussed several representations, namely voxels, implicit surfaces, FRep, RBF, SDF, ADF and IDF, as well as some hybrid approaches that are suitable for heterogeneous volumetric modelling. Voxels can be used for defining various volumetric shapes with different attributes specified per voxel, but this representation is discrete and essentially approximates the geometry. On the contrary, FRep is at least C^0 continuous, the objects are defined by the functions that means that they can be defined with nearly infinite precision, and it is already provides different hybrid properties. Unfortunately, it is problematic to render and it has a non-distance based unpredictable behaviour. The advantages and drawbacks of FRep can be found in table 2.4.

SDFs, ADFs and IDFs provide different distance fields for the defining objects. They provide a distance based predictable field behaviour for the defining object and can be easily rendered. However, they also had their drawbacks that can be found in tables 2.5, 2.6 and 2.7.

From the conducted review, it follows that FRep might serve as the basis for constructing a hybrid representation that will unify advantages of FRep, SDF, ADF and IDF and compensate for their drawbacks. In the next sections we will review the state-of-the-art methods for defining attributes in volumetric objects to finally formulate the hybrid framework for multi-material heterogeneous object modelling.

2.4 Multi-Materials and Microstructures in Heterogeneous Modelling

Attribute distributions specified in heterogeneous objects O_H can be uniform or non-uniform. For instance, the simple example of the uniform distribution can be a homogeneously coloured object. As to non-uniformity, it can be presented as porous structures or microstructures with non-linear varying density or multi-material distributions. In this section, we discuss the state-of-the-art on how multi-materials, microstructures and porous structures can be efficiently represented in heterogeneous objects.

2.4.1 Methods for Defining Multi-Materials in Heterogeneous Objects

A material can be represented as a composition of atoms of several chemical elements within the given volume. Atoms can be combined in different structures like molecules, particles, grains, crystals and other, forming microstructures on several scale layers. Materials can be characterised by its concentration or density representing the certain fraction of a single or multiple materials (Schmitz et al. 2016).

Generally, there are three classes of multi-material representations that are widely used: composite materials, functionally graded materials (FGMs) and multi-materials that are defined using additive manufacturing techniques. Composite materials can be represented as a combination of multiple homogeneous materials with different physical properties sharing the boundaries within an object. FGMs are composites that consist of two or more materials that are gradually changing with position or direction (Niino et al. 1987). Additive manufacturing techniques (Ngo et al. 2018) provide methods for a digital fabrication of multi-material heterogeneous volumetric objects using 3D printing. Multi-materials along with the geometry are defined using voxels for fusing them together at the given volume element (Bader, Kolb, Weaver, Sharma, et al. 2018; Brunton et al. 2018; Cimquest and Hewlett-Packard Development Compahy 2014; Doubrovski et al. 2015) or for mixing several materials together using a dithering technique (Stratasys 2014).

Multi-material heterogeneous volumetric objects consist of two elements. The first element is a geometric shape of the object that can be defined using BRep, any solid representational scheme or a scalar field (Biswas, Shapiro, and Tsukanov 2004; Pasko, Adzhiev, Schmitt, et al. 2001). The interior of the object can be defined as domains, partitions or cells sharing their boundaries. The second element is a material distribution which can be set up in several different ways.

The material distribution can be defined by assigning a material index as an integer value (Kumar and Dutta 1997) that is suitable for the composite materials definition. Material distributions can be set up in a discrete manner storing its values at each cell of the data structure, e.g. voxel (Hoetzlein 2016; Museth 2013). They can also be defined using piecewise polynomials (Yuan, Yu, and Wang 2012) or continuous scalar fields (Biswas, Shapiro, and Tsukanov 2004), which provide a resolution independent material distribution or nearly infinite resolution controlled by computer precision. There are several interesting examples discussed in (Biswas, Shapiro, and Tsukanov 2004). In particular, the distance-based smooth and differentiable attribute functions were applied to represent a parabolic distribution of the graded refractive index in Y-shaped solid of the waveguide as it can be seen in Fig. 2.17. In this case, it is important that the distribution of the index of refraction is

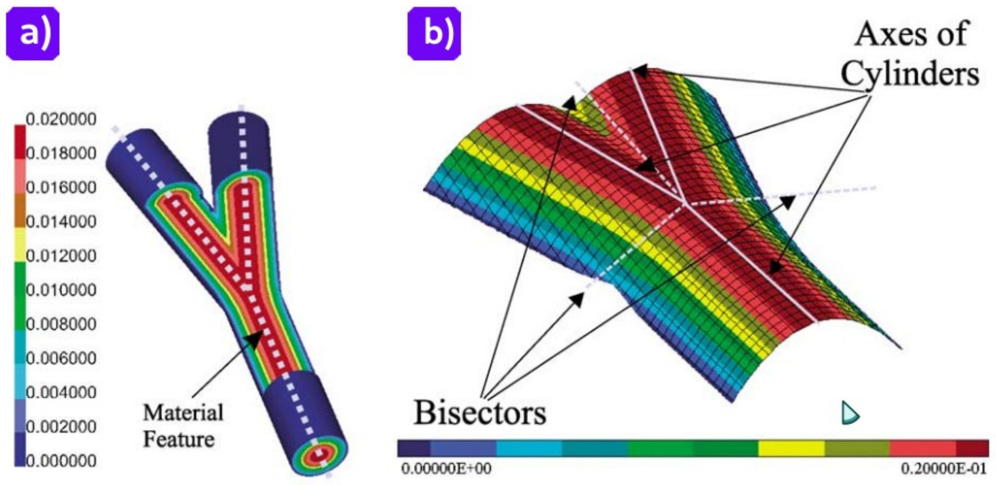


Figure 2.17: *Refractive index distribution of a radial Y-branching waveguide. a) Refractive index distribution specified for the Y-shaped one-dimensional tube; b) Magnified distribution plotted as a surface over a 2D section. Reprinted from (Biswas, Shapiro, and Tsukanov 2004), Copyright ©2004, with permission from Elsevier.*

uniform and smooth.

The Scalar Field Based Approaches

Piecewise continuous scalar fields are sufficient for defining a resolution independent material distribution within a geometric shape of the object. One of the most common approaches is to make a spatial partitioning of the object beforehand. Then unique material distributions in obtained spatial regions are defined. The voxel representation (Kaufman 1994; Wang et al. 2011) (see subsection 2.3.1) for the volumetric objects is the most widely used approach, especially in case of objects that are further used for 3D printing (Bader, Kolb, Weaver, and Oxman 2016; Bader, Kolb, Weaver, Sharma, et al. 2018). In Fig. 2.18 we show an example of multi-material heterogeneous 3D printed object (a model of human brain), which was defined using voxel representation (Bader, Kolb, Weaver, Sharma, et al. 2018). The model demonstrates the bundles of axons in the brain with various connections. However, voxel representation has several significant restrictions that were highlighted in table 2.2.

In (Kumar, Burns, et al. 1999) the geometric shape of the object was defined by a topological cell complex with attributes defined per point with C^k continuous scalar fields. The attribute at each point represents a material property (e.g colour) or a material density. In (Bhashyam, Shin, and Dutta 2000) that approach was further generalised. The authors defined several analytical material composition functions that were used for the definition of the material independent of geometry.

Another variant of setting up material properties was suggested in work (Pasko, Adzhiev, Schmitt, et al. 2001). A hybrid representation on the basis of FRep called constructive hypervolume model was introduced. In this work object properties were set up in a point-wise manner. The authors (Sharma and Gurumoorthy 2017) suggested to use the decomposition of the geometry that relies on the existing class of material distance-based functions. These functions were applied for the definition of a material variation in heterogeneous objects using the medial axis transform.

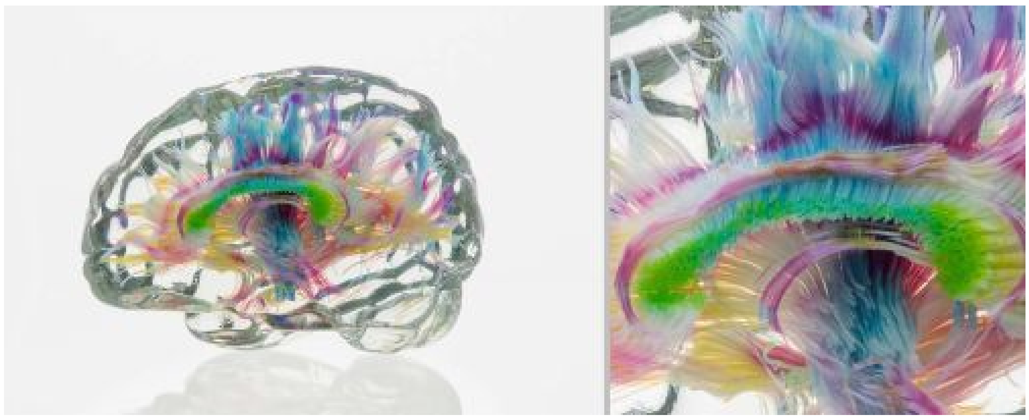


Figure 2.18: *White matter tractography data of the human brain, created with the 3D Slicer medical image processing platform, visualising bundles of axons, which connect different regions of the brain. Reprinted from (Bader, Kolb, Weaver, Sharma, et al. 2018), Copyright ©2018, the Authors, some rights reserved; exclusive licensee American Association for the Advancement of Science* [a].

[a]Creative Commons Attribution 4.0 International License (`http://creativecommons.org/licenses.by/4.0/`).

Procedurally Defined Materials

In some cases, material distributions can be described in the closed form of an analytical expression over the point coordinates. These expressions can be defined as the pure polynomials or more general functions, e.g. trigonometric, exponential and others. The authors (Kumar and Dutta 1998) suggested to use the power law in the form of the polynomial function for describing multi-materials. In another work (Gupta, Kasana, and Tandon 2010) logarithmic functions were suggested to use for defining the material distribution in the object. The material function was defined as a distance function from the end point of the first homogeneous region to the first differential geometric point. In work (Bhashyam, Shin, and Dutta 2000) was proposed a variety of different composition functions including polynomial, exponential and trigonometric. In (Yang et al. 2014) was suggested to use Gaussian radial basis function for interpolating between boundaries of the regions where materials were defined. The resulting model is C^∞ continuous and can handle complex structures.

The Feature-Based Approaches

The feature-based or source-based class of methods is another approach for defining attributes in interior of the object. An object is partitioned into several regions and in selected regions an attribute is assigned using a distribution function that creates a source of gradient material (Biswas, Shapiro, and Tsukanov 2004; Park, Crawford, and Beaman 2000; Qian and Dutta 2004). The materials are blended together using specified weights that are proportional to the distance to a feature. In (Siu and Tan 2002a,b) a source-based approach was introduced. The attribute distribution was defined using grading sources. These sources can be represented as points, lines, and planes arbitrary placed in a space or at the surface boundary. The material composition at the given point is evaluated in accordance with the distance from the current point to each source.

Summary

In this subsection we have discussed several classes of methods for defining attribute distribution in interior of the object. There are some promising methods for attribute definition that preserve continuity of the geometric shape and attribute distribution in interior of the object. This class of methods utilises the functional nature of scalar fields, particularly, distance-based. The parameterisation of the attribute functions by the distance provides intuitive control for the attribute definition.

In this work we would like to consider the parameterisation of the attribute functions by at least C^1 continuous smooth distance functions as we want to avoid the appearance of the stresses and creases in the modelling attribute. By applying such parameterisation, we will be able to introduce a novel class of the smoothed out attributes with intuitive control.

2.4.2 3D Printing

Polygonal representation is widely used in 3D printing. The industry standard file format for 3D printing is STL that was introduced by 3D Systems company (3D Systems 2020) in 1986. It is a simple data structure that could handle only the description of the geometric shape. According to (Hiller and Lipson 2009), the STL file format contains information only about a surface mesh and it is not capable to represent colour, texture or any material. The geometry is stored as an unordered list of triangles. As each triangle is stored separately, each vertex might be presented repeatedly for every triangle. This leads to leaks in the defining geometry, where small rounding errors results in vertices that are misaligned. Generally, this format cannot be used for 3D printing of heterogeneous objects without extra steps (e.g. represent each material by its own STL file (Lei et al. 2014)) or extensions of the format.

In (Hiller and Lipson 2009) a revision of the STL format was proposed. The modified version of the STL format was suggested to name additive manufacturing file format (AMF). The AMF file format can represent one or multiple objects that are arranged in a constellation. The geometric shape of the object is still defined as a triangular mesh. The main advantage of the AMF format over the STL format is that it is able to store the information about the colour and materials specified for the object. Material distributions can be set up for the object using FGMs defined by analytical functions of point coordinates. Relatively recently, the ISO organisation (Subcommittee 2020) took a decision to replace the STL file format with AMF. Unfortunately, it still provides no validation of triangular surfaces.

It was developed several alternatives to STL and AMF formats for 3D printing that cover multi-material support (Qin et al. 2019) and most of them represent the geometry using BRep. For instance, 3MF (3D manufacturing format) (3MF 2020) format developed by Microsoft defines the geometry as a triangular mesh. This file format is capable to store additional information about material and textures defined for the object.

The most suitable way for defining heterogeneous objects in the context of 3D printing is to use voxel based or similar data formats. For example, in (Kou, Tan, and Sze 2006) was proposed a novel data structure based on non-manifold cellular representations. This approach was able to define complex, smooth and versatile material distributions upon the geometry. A voxel based method for layered manufacturing was first introduced in (Chandru, Manohar, and Prakash 1995), where several core problems were addressed such as slices generation, estimating surface

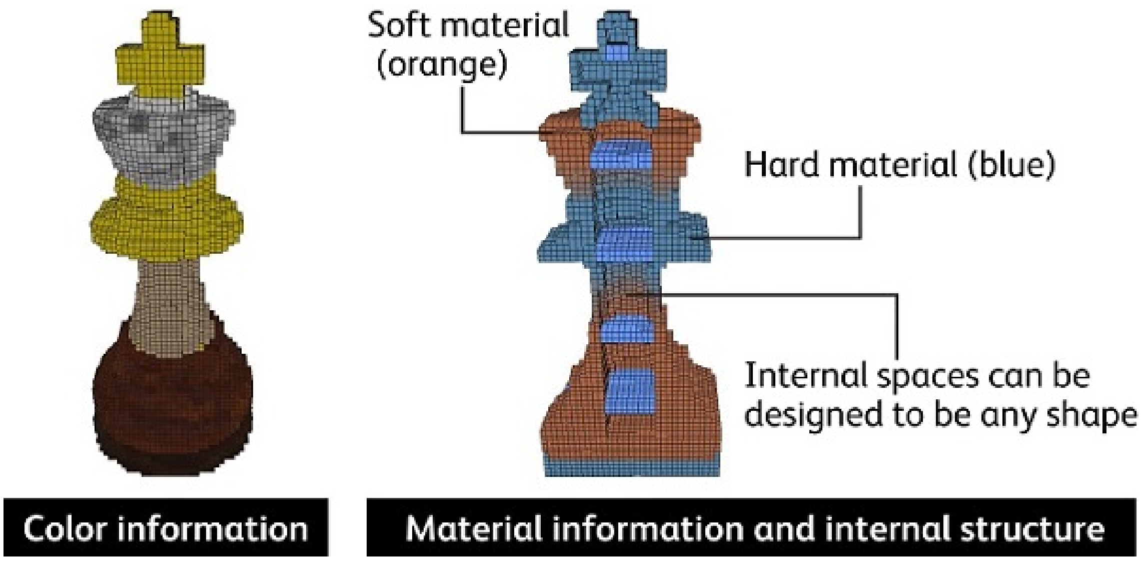

Soft material (orange)
Hard material (blue)
Internal spaces can be designed to be any shape
Color information
Material information and internal structure

Figure 2.19: *The illustration of FAV 3D printing format that is suitable for printing heterogeneous multi-material objects and was proposed by Fuji Xerox company (Takahashi et al. 2018). Reptinted from (Takahashi et al. 2018), ©2018, Fuji Xerox Co., Ltd. and Dr. Hiroya Tanaka* [a].

[a]Creative commons, Attribution, No Derivative Works, 4.0 International (`https://creativecommons.org/licenses/by-nd/4.0/`)

properties, etc. In (Doubrovski et al. 2015) a bitmap 3D printing method with multi-material support that is also voxel-based was proposed. Quite recently, the research scientists from Fuji Xerox have introduced a new format FAV (Fabricatable voxel) (Takahashi et al. 2018) that allows users to design a complex distribution of attribute values or internal structures to be 3D printed using voxel data structure. In Fig. 2.19 we show a picture from the report by Fuji Xerox that describes how multi-materials with different properties can be stored using FAV format.

2.4.3 Methods for Microstructure Generation

As it follows from the definition 2.1.1 of a heterogeneous object, the interior part of such an object can be represented with different complex structures such as microstructures or porous structures. The interior structures can be used for providing additional durability to the object that will be 3D printed with less material. They can be important while computing dynamic simulations or for replicating the interior structure of the real object. In computational material engineering, such structures as microstructures are considered as the carrier of material properties and its density distribution, which are derived from the features of the nano-scale and micro-scale level of detail.

We can highlight two major types of heterogeneity depending on the scale level of detail. If the material is defined as a non-uniform material density distribution through the volume on the nano-scale level, then it is a *compositional heterogeneity*. Otherwise, if the material is defined as a variable spatial structures, e.g porous structures, lattices, scaffolds etc., on the micro-scale or meso-scale levels, then it is a *structural heterogeneity*. Both types are widely used in the heterogeneous object modelling (Markworth, Ramesh, and Parks 1995; Regli et al. 2016; Schmitz 2016).

The authors (Siu and Tan 2002b) introduced a method for modelling grading

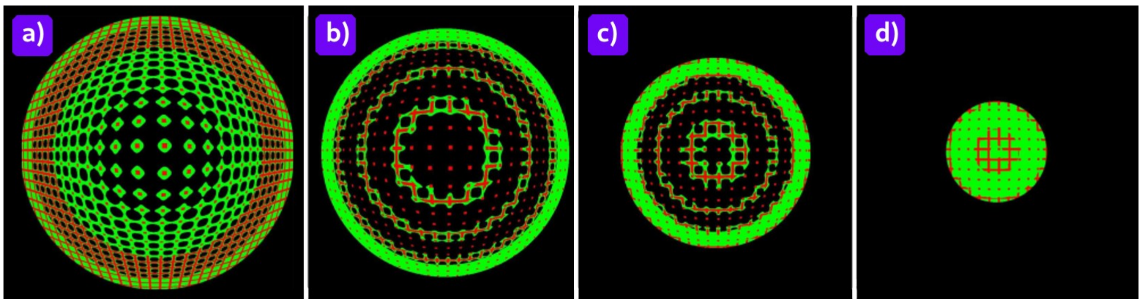


Figure 2.20: *Multi-material FRep object with interior microstructures in the form of the rods. One material was defined for the rods and another material was added using blending operation. Reprinted from (Pasko, Fryazinov, et al. 2011), Copyright ©2011, with permission from Elsevier.*

materials and interior structures of heterogeneous objects. The interior structure of the object was defined using structure arrays that contained information about both material composition and geometric structure of the object. In work (Schumacher et al. 2015) a method to define and preserve a spatially varying elasticity property for the 3D printed objects with defined microstructures was presented. Microstructures were specified using L_2-distance fields normalised in the interval $[0, 1]^d$ that corresponds to a metamaterial space. To avoid discretization errors each microstructure sampling from the metamaterial space was smoothed using Gaussian smoothing step. In work (Pasko, Fryazinov, et al. 2011) a novel approach for modelling microstructures and irregular porous media using FRep was presented. The described FRep-based method was used for constructing volumetric structures with the size of the details that is orders of magnitude smaller than the overall size of the object. The introduced method for generating microstructures is also suitable for multi-material object generation as it can be seen in Fig. 2.20. In another work (Fryazinov, Vilbrandt, and Pasko 2013) the proposed method for generating microstructures using FRep was generalised for the construction of the cellular structures. The geometry of a base unit cell was defined using FRep. Then, by applying a periodic space mapping, this unit cell was multiplied over the object interior.

Microstructures can also be constructed using a tiling of the deformation domain (Massarwi et al. 2018). The resulting tiling is stored as a deformation map that provided the mechanism for decoupling of the micro and macro structures. A particularly interesting case of using microstructures was described in (Auzinger, Heidrich, and Bickel 2018). In this work microstructures were used for modelling nanostructural colour using additive manufacturing techniques. The structural colours were designed as a combination of the following methods: a full-wave simulation, a proper parameterisation of the design space and a tailored optimisation procedure.

Porous structures are a particular sub-class of microstructures that is challenging to generate due to their high degree of irregularity and intricacy of their geometrical structure. As it was suggested in work (Kou and Tan 2010), they can be generated using a colloid-aggregation model. The authors used Voronoi tessellation for partitioning the space into a collection of regions, which were merged together to imitate the random colloid aggregations. An example of the generated models using this method can be seen in Fig. 2.21. An alternative approach was suggested in (Yoo 2012) where porous structures were defined using implicit solids, SDFs and radial basis functions (RBF). These representations were used for modelling a minimal surface porous scaffold. The researchers from Autodesk Inc. (Michalatos and

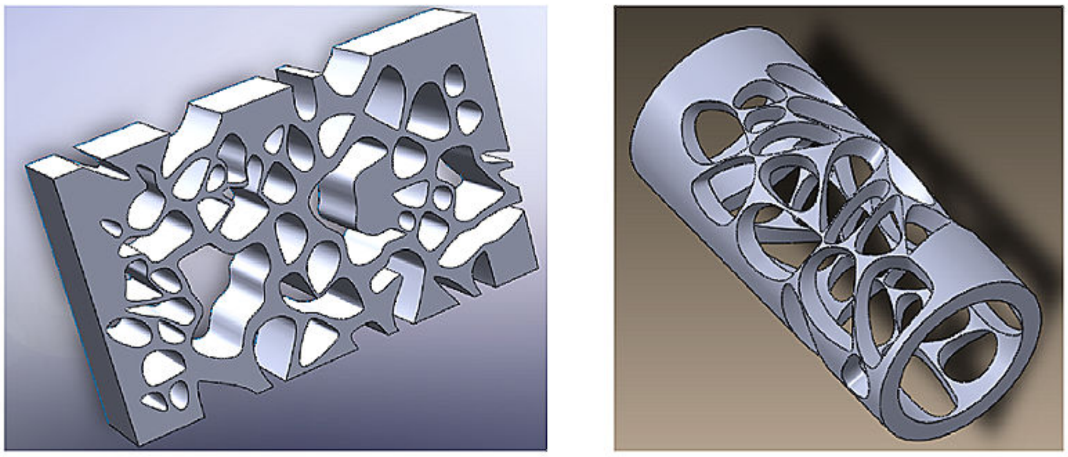


Figure 2.21: *Example of the 3D porous structures generated using method described in (Kou and Tan 2010). Reprinted from (Kou and Tan 2010), Copyright ©2010, with permission from Elsevier.*

Payne 2018) suggested a modulated binarisation method that combined volumetric texture coordinates with micropatterns halftonic techniques, which were defined by potential fields of the texture coordinates. The obtained structures were specified across multiple scales with assigned material properties.

Summary

Interior structures play an important role in heterogeneous volumetric modelling. They can be defined at different scale levels and can be potentially specified as a carriage of various attributes. The presence of the interior structures is important for additive manufacturing to provide additional durability for the 3D printed objects.

As it follows from this subsection, there are a lot of different methods for defining micro-structures and porous-structures. For instance, they can be conveniently defined using continuous functions, e.g FReps. However, FReps provide a non-distance based field as we have discussed in subsection 2.3.3. Thus, it is hard to use such objects in physically based simulations, 3D modelling and 3D rendering without conversion to another representation. However, if FRep can be somehow combined with distance-based representations, it is possible to obtain a convenient and predictable way for defining various interior structures defined in heterogeneous objects.

2.5 Dynamic Heterogeneous Objects

Heterogeneous objects are widely used in different physically-based simulations, where the surface of the object as well as its interior structure participate in the simulation process. Another application of volumetric objects is related to various morphing tasks. In this section we review works dedicated to feature-based and automatic morphing techniques as these methods deal with time variant objects.

2.5.1 Feature-Based Morphing

In this subsection, we consider non-automatic methods for morphing with additional user control. An early example of an in-between metamorphosis for 2D images can be found in (Burtnyk and Wein 1976) where the authors present a system for morphing between hand-drawn contour shapes with in-between shape generation. The in-between interpolation relies on establishing correspondences between the shapes along with stroke to stroke mapping. To fully utilise the system, a skeleton must be set up. A similar method was presented in (Eisemann et al. 2008), where pixel tiles are moved from one image to another according to the optical flow while changing their colours.

Some attempts to introduce more flexible user control over morphing were undertaken with user-defined feature segmentation. A feature-based image metamorphosis technique was presented in (Beier and Neely 1992) for obtaining smooth transitions between two images. This allows the user to define the pairs of corresponding features in the source image and the target image that are further used for computing in-between frames. This, however, heavily relies on the linear interpolation between shapes which does not produce acceptable results in most cases. Higher quality interpolation with provided corresponding boundary point pairs is described in (Alexa, Cohen-Or, and Levin 2000).

The simplest technique for transforming one digital image into another in terms of in-place morphing (when the input and target shapes are superimposed on one another) is cross-dissolving (Smith 1987) where one image gradually disappears and another appears through a per-pixel colour interpolation. This method can only handle image morphing. According to (Lee, Wolberg, and Shin 1998; Wolberg 1998), cross-dissolving without pre-warping the initial and target shapes, produces a poor result with a double-exposure effect. For obtaining best results two images should be pre-warped to make the shapes similar by specifying control pixels. This leads to the non-automatic approaches described, for example, in (Beier and Neely 1992; Smith 1987; Wolberg 1998).

Dalstein et al. (Dalstein, Ronfard, and Panne 2015) introduced a feature-based method using the Vector Animation Complex, which is a vector graphics data structure. Their approach uses the space-time concept that is based on a parameterised model for obtaining in-between frames. The suggested method is topology-aware, can work with overlapping objects and supports colouring of the 2D faces. Unfortunately, this method can produce discontinuous results because of the employed linear interpolation and it also does not support colour/texture transformation between frames.

2.5.2 Automatically Controlled Morphing

Averbuch-Elor et. al. in (Averbuch-Elor, Cohen-Or, and Kopf 2016) introduced a data-driven method based on the inner-distance shape context technique to handle geometry transformations and globally affine RGB transformations for colour blending. Their in-between images are created by warping and blending an input image toward a target image to align them. This method cannot handle objects with different topology, and also does not take into account the interior texture of the object.

Gao et. al. introduced a data-driven mesh morphing method in (Gao, Chen, et al. 2017). Their method depended on patch-based and linear rotationally invariant coordinates that can handle the deformations of models in a shape collection. This method works with objects of the same topology and does not handle texture or colour transformations.

Liao et. al. (Liao et al. 2014) proposed an automatic method for optimisation-guided image morphing that also supports the presence of greater dissimilarities between images. The optimisation can take into account user-drawn points for better aligning of image features. While providing intuitive results, this work does not provide a method for dealing with objects of radically different topology.

A hybrid stroke-based solution (Jin and Geng 2015) deals with automatic generation of in-betweens for 2D facial cartoon animations. The key idea of the suggested hybridisation is to combine the information about the approximate 3D geometry with the multiple views of a character's face from key frames to overcome the lack of information. It facilitates automatic establishing of stroke correspondences. However, the strokes can be incorrectly annotated, and this method cannot handle morphing between radically different images such as drawings with dissimilar strokes or shapes.

An interesting approach was introduced by Neumann et. al. in (Neumann, Alexander, and Neumann 2017) where a random walk algorithm was applied for obtaining evolutionary image morphing. The proposed algorithm is based on the usage of fitness functions for a per-pixel image transformation using random walks and is only texture aware.

The most relevant group of methods for automatically controlled metamorphosis is related to optimal mass transport (OMT). The OMT methods provide a solution for the Monge-Kantorovich optimisation problem (Kantorovich 1948). This solution provides the optimal way for moving a mass distribution from one domain to another with minimal transportation cost. Typically the solution is obtained in the form of the L^2 Kantorovich-Wasserstein distances by solving the differential equation defining the metric. In the context of the metamorphosis method it is important to note that OMT-based methods are parameter free and not feature-based.

In (Haker et al. 2004) and (Zhu et al. 2007) an intensity-based OMT method was introduced. The process of obtaining the in-between frames assumes computation of the Kantorovich-Wasserstein distances which are used for generating the warping transportation map between the initial and target images. Then the cross-dissolving method is applied. In (Zhu et al. 2007), to overcome the double-exposure effect caused by cross-dissolving, the authors introduced an intensity penalty term to the mass moving energy functional.

A vector-valued OMT method was introduced in (Chen, Georgiou, and Tannenbaum 2018). This method allows handling of the mass flow between vectorial entries across a discrete or continuous space. The authors claimed that their approach is suitable for a number of applications, in particular for colour image processing and for morphing between colour distributions. They have provided an example of colour interpolation for real-life images in the form of photos that can be seen in Fig. 2.22. However, this is a pure image processing example without taking into account geometry.

The variational OMT approach (Maas, Rumpf, and Simon 2017) was applied to grey-scaled textures with sharp features. In (Makihara and Yagi 2011) a method for topology-aware shape morphing using cluster-based earth mover‘s distance flows was introduced. Unfortunately, it cannot handle any colour or texture morphing. The author (Lévy 2015) introduced a numerical topology-aware method for dealing with geometric metamorphosis only. It is based on computing OMT between density in the form of a piecewise linear function and a sum of Dirac masses. The

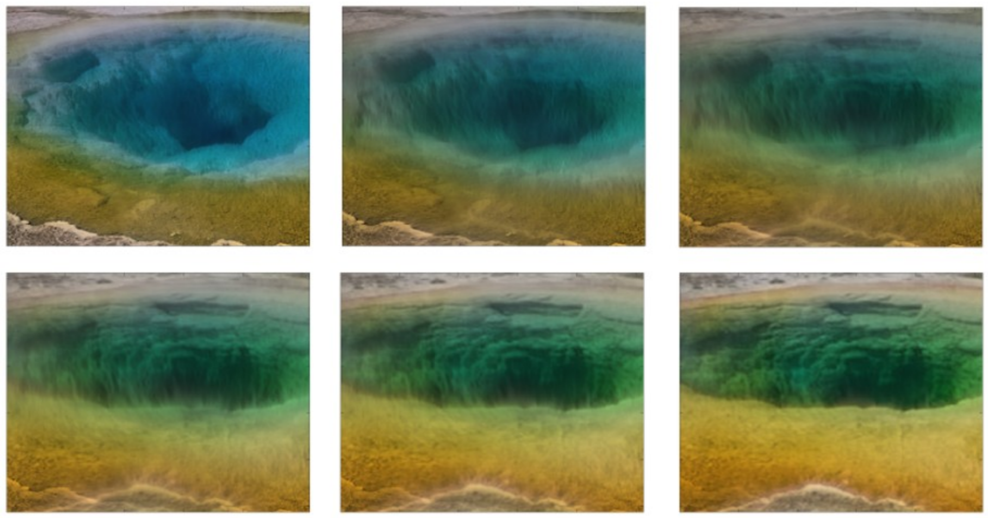


Figure 2.22: *Example of the OMT-based morphing method described in (Chen, Georgiou, and Tannenbaum 2018) that interpolates corresponding vector-valued distributions. Reprinted from (Chen, Georgiou, and Tannenbaum 2018). Copyright ©2020 Society for Industrial and Applied Mathematics. Reprinted with permission. All rights reserved.*

authors in (Solomon, Goes, et al. 2015) and (Cuturi and Peyré 2016) introduced a topology-aware method for solving the OMT optimisation problem using convolutional Wasserstein distances that are approximated using entropic regularisation. This method can only handle interpolation between blocks of pure colour.

In (Rehman, Pryor, and Tannenbaum 2007) a GPU based approach for solving the OMT optimisation problem was suggested and applied to high frequency grey-scaled images. Elsewhere the authors (Nader and Guennebaud 2018) introduced a new solver for computing an approximated OMT that is derivative free and converges within a few iterations. This method is topology-aware, preserves sharp features during metamorphosis but does not support texture or colour transformations.

Volumetric Morphing Techniques

Most known solutions for volumes or scalar distance field representations work only with geometry, and only few can handle both geometry and attribute transformations. Let us mention only the most representative related works here.

Breen and Whitaker (Breen and Whitaker 2001) introduced a level-set method for representing the deformable surface of a densely sampled scalar function. A colour interpolation is implemented using a trilinear interpolation combined with the scan-conversion closest point method, where, unfortunately, the trilinear interpolation only handles simple colour transformations. Dinh et. al. (Dinh et al. 2005) introduced a PDE based method that could transfer textures during shape transformation. Their method is based on solving the Laplace equation for defining flow lines that execute the bijective transformation between input and target objects. For texture mapping they used the Laplace equation with the Laplace-Beltrami operator to establish pointwise correspondences between two objects defined by implicit functions raised into 4D space.

Weng et. al. (Weng et al. 2013) introduced a metamorphosis method for objects that are defined using distance fields that exploits a OMT-based method that pro-

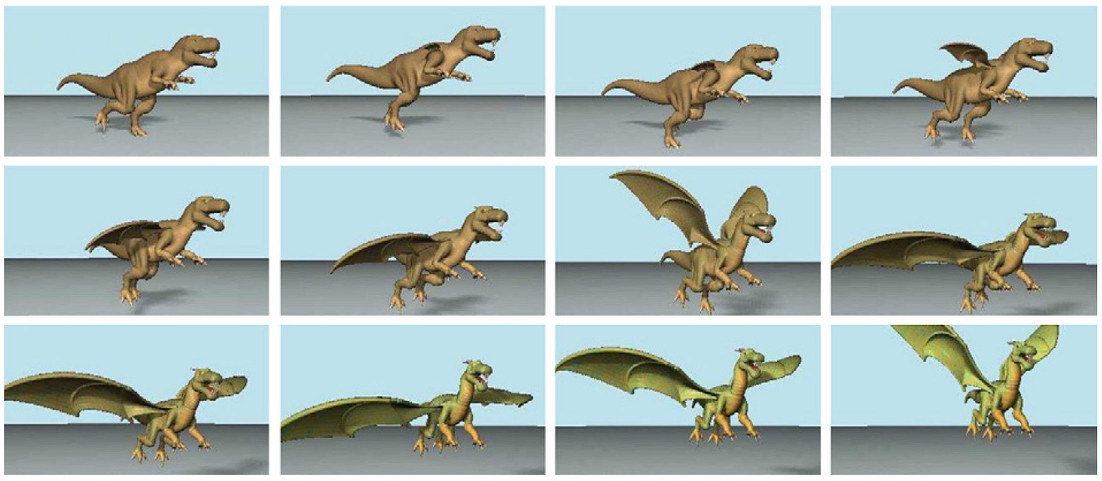


Figure 2.23: *An OMT-based metamorphosis between a walking Tyrannosaurus-Rex and a flying Dragon. Reprinted from (Barbier, Galin, and Akkouche 2005), Copyright ©2005, with permission from Elsevier.*

vides the optimal way for moving a mass distribution from one domain to another with minimal transportation costs. For better alignment of initial and target objects during metamorphosis, the method relies on establishing correspondences between two objects that are used for computing a non-rigid warping function. Unfortunately, this method is quite sensitive and cannot handle colour or texture transformations.

PDE based methods are in most cases computationally expensive. There are also methods that are less expensive to compute but they usually produce less satisfactory results. Barbier et al. (Barbier, Galin, and Akkouche 2005) introduced a metamorphosis operation for textured objects using the BlobTree model where initial and target objects are represented in 4D space. Geometry and texture transformations are applied simultaneously using a warping function for geometry and one of their blending functions for textures. In Fig. 2.23 we show how this method was applied to two animated models. The accuracy of the shape transformations was achieved using additional controls. Another method for metamorphosis between textured objects uses the unstructured lumigraph representation (Ludwig et al. 2015). The texture transformation was achieved using a simple linear interpolation that influenced the output quality.

Summary

As it follows from this section, methods and representational schemes for dynamic heterogeneous objects are not well developed yet. We have reviewed works dedicated to various feature-based and automatically controlled morphing and metamorphosis methods. Only feature-based methods could efficiently handle geometry and attribute transformations of any complexity simultaneously and interconnectedly. However, there is lack of automatically controlled methods that could be applied to heterogeneous objects with complex attributes (e.g textures, materials).

Many of the described methods for a 2D textured metamorphosis work reasonably well with pure image data (raster arrays or textures) rather than with textured 2D shapes. In the reviewed works we show that there is lack of methods that could handle a heterogeneous metamorphosis automatically and without establishing any correspondences between input and target objects. There were only several methods

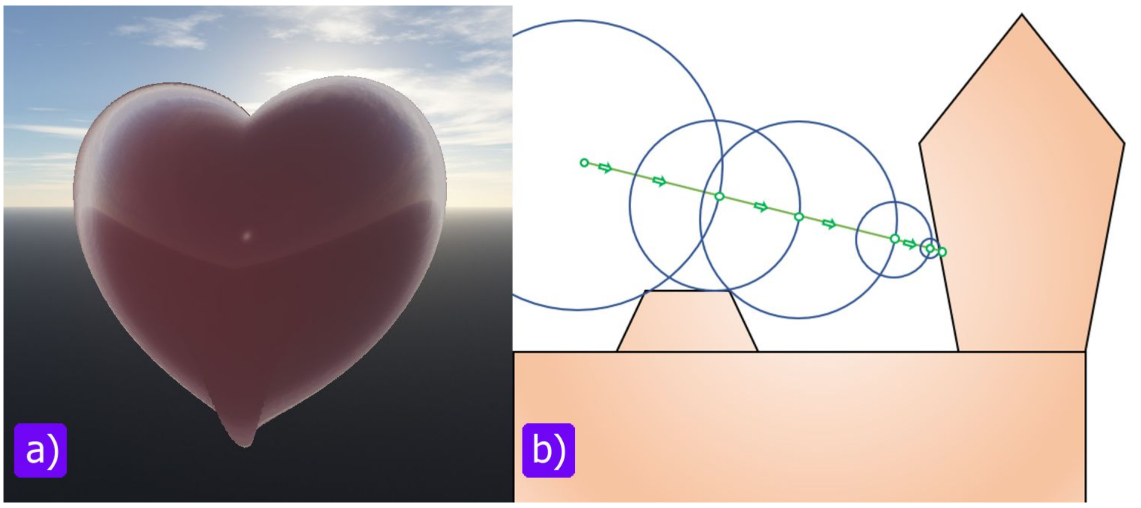


Figure 2.24: *The result of the sphere-tracing algorithm a), and its concept b).*

that could handle either metamorphosis between homogeneously coloured objects or between two colourful pictures. The most relevant method in the context of automatically controlled metamorphosis is OMT. This class of methods is parameter-free and topology aware. However, the reviewed works did not provide the implemented examples of metamorphosis between objects with sophisticated textures.

We can see that among 3D metamorphosis techniques there is also lack of methods that could handle both geometry and attribute transformations simultaneously and interconnectedly. In case of OMT-based methods, establishing correspondences can be important for better alignment of the initial and target objects.

2.6 Rendering of Multi-Material Heterogeneous Objects

There are several well-established groups of methods that are used for rendering heterogeneous volumetric objects. They are marching cubes based algorithms (Wu and Sullivan Jr 2003), volume ray marching based algorithms (Hart 1996; Keinert et al. 2014), splatting (Westover 1990; Westover 1991), shear-warp based techniques (Lacroute and Levoy 1994), and texture-based algorithms (Fernando 2004).

The marching cubes method (Lorensen and Cline 1987) is used for converting the volumetric object to the polygonal mesh that can be rendered using any convenient graphics pipeline. While defining the position of the object surface, the algorithm is searching for the intersection of the surface with the current cube. Then it defines which of the triangles referring to the inside (set to zero) and outside (set to one) of the object. After all sequential traversals of the cubes, the triangulated mesh is obtained and surface normals are defined. The marching cubes algorithm was also extended for rendering volumes with defined multi-materials (Wu and Sullivan Jr 2003). In the suggested algorithm, boundary surfaces between different materials were extracted in one sweep of the image stack. The resulting object contains no voids or overlaps between materials, and each material was bounded by the triangulated surface.

In (Drebin, Carpenter, and Hanrahan 1988) a novel technique for rendering volumetric objects represented as voxels taking multi-materials into account was suggested. First, the input data volume is converted to material percentage volumes which were further used for composite volume generation. Then, to define surface normals and boundaries between materials, the gradient operator was applied to the density. The obtained object preserved the continuity of the input data volume which was represented as a sampled continuous signal.

Another well-developed class of algorithms for volume rendering is the ray marching based techniques which are widely used for rendering SDFs and ADFs. The main idea of these methods is to find the intersection between the view ray initialised at the camera view position and the scene with an object. In (Hart 1996) a ray marching based algorithm called sphere tracing was proposed. The main goal of the suggested technique was to render implicit surfaces or scalar fields, which could be creased, rough and even defined by functions with discontinuous or undefined derivatives. The concept of the sphere tracing can be described as follows (see Fig. 2.24, b). For the given object defined by SDF, at each position along the view ray (Fig. 2.24 (b), red line) the distance in the current point is treated as a radius of the sphere centred around the current position. If the sphere does not contain any surface, the step along the ray is set to be equal to the current sphere radius. In (Keinert et al. 2014) an enhanced sphere tracing algorithm was described. Several techniques for accelerating the rendering process, enhancing the intersection calculation between the ray and the surface, and reducing discontinuity artefacts were proposed.

In work (Igouchkine, Zhang, and Ma 2018) a method for rendering multi-material volumes taking into account the physically-based surface reflection function was introduced. The volume rendering transfer function was extended for supporting a surface-like behaviour at the boundaries between volume components and volume-like behaviour inside the volume. They implemented their method using the ray marching algorithm.

Summary

In this section we have discussed the core algorithms for rendering heterogeneous objects that are defined using scalar fields. One of the slowest methods is marching cubes that was also applied to the multi-material heterogeneous objects. Ray marching based methods could suffer from an over-shooting problem if the step along the ray is chosen not sufficiently small. On the contrary, a sphere-tracing method fully utilises the nature of the distance fields. It could handle discontinuities and other artefacts, and it is efficient for obtaining interactive rendering rates. This method can also be applied for rendering multi-material objects taking into account different physically-based properties of the attributes.

2.7 Conclusions

In this chapter we have discussed the core representations that are suitable for defining multi-material heterogeneous objects. The boundary representation (BRep) is one of the most widely used, but least suitable for volumetric modelling as it is surface-based, hollow inside, discrete (if it is polygon-based) and essentially approximates the surface of the object. It is still a prevailing representation, particularly, in 3D printing, where the presence of interior structure is omnipresent, as most of the supported formats are STL-based. In the case of multi-material 3D printing, this format is cumbersome. One of the perspective directions is to develop voxel-based formats. As it was discussed in subsection 2.4.2, Fuji XeroX have already introduced a new format called FAV.

As in this work we are dealing with multi-material heterogeneous objects, we consider volumetric representations as the basis for the hybrid framework that will be presented in chapter 3. We have reviewed several of them, namely implicit surfaces, voxel representations, FRep, SDF, ADF and IDF, with identified advantages and drawbacks. In volumetric modelling, the most widely used representation is voxels due to their simplicity. However, as it was discussed in subsection 2.3.1, voxels are not efficient in terms of computing as they might consume a lot of memory. This problem can be solved using various adaptive approaches. Nevertheless, this representation is discrete and essentially approximates the geometry and specified attributes in interior of the voxelised object. The accuracy of the voxelised geometry and attributes depends on the voxel grid resolution.

On the contrary, scalar-based fields are continuous (except ADFs) and the accuracy of the volume definition with specified attributes depends on the computational precision. These representations can be efficiently processed, rendered and stored. They can be converted to BRep or voxel representations that are widely used in industry.

FRep is a representation that generalises implicit (surfaces) modelling and extends a CSG approach. The FRep defining function is at least C^0 continuous, but it can be enforced to be C^1 continuous. There are a lot of well-established operations for FRep. It can be used for modelling microstructures, cellular structures and porous structure by utilising its functional nature. Unfortunately, FRep is not distance-based and, as a consequence, it can sometimes be problematic to render. Some operations defined for FRep produce an unpredictable field.

On the other hand, distance-based representations (SDF, ADF and IDF) provide a predictable and intuitive control for the geometric and attribute modelling. They can be efficiently handled and rendered. However, each of the reviewed representations, namely SDF, ADF and IDF, have their drawbacks. SDFs are not smooth and might not be differentiable everywhere, might have discontinuous gradients on the medial axis and according to (Biswas, Shapiro, and Tsukanov 2004) are not suitable for attribute modelling as they are C^0 continuous. ADFs have C^0 and C^1 discontinuities and are also not suitable for the attribute modelling. IDFs are defined only in the interior of the object.

The discussed representations have a lot of fruitful advantages that can be found in tables 2.4, 2.5, 2.6 and 2.7 that we would like to preserve and compensate for their drawbacks. According to subsection 2.3.8, a possible solution to achieve this goal is to construct a hybrid framework.

According to (Gómez et al. 2019), fast methods for solving the eikonal equation have a variety of different applications. They can be used in path planning in robotics (Do, Mita, and Yoneda 2014; Gao, Wu, et al. 2018), image segmentation (Forcadel, Guyader, and Gout 2008), volumetric data representation (Museth 2013), visual effects and others. However, they are not always efficient to compute, especially if the implementation of the method is not paralleled. In this case, using adaptive Cartesian grids (e.g. T-meshes, quadtrees, octrees etc.), it is possible to reduce the amount of computations. The FIM method seems to be the best candidate to be adopted for computations on adaptive grids as every node of the grid can be updated independently.

One of the stated research problems in this work is dedicated to developing dynamic methods for heterogeneous volumetric objects. Particularly, we are interested in morphing and metamorphosis techniques. In section 2.5, we have reviewed different works that are dedicated to solving the metamorphosis problem (both in 2D and 3D) of one object into another. We have identified that there is lack of methods that could handle both geometry and attribute transformation simultaneously and interconnectedly without establishing any correspondences. We will show in chapters 5 and 6 how to solve the automatically controlled metamorphosis problem without establishing any correspondences in 2D and 3D.

According to section 2.6 there are a lot of methods for rendering heterogeneous volumetric objects. Among all reviewed volumetric rendering techniques, the most well-suited for distance-based objects is the sphere-tracing method. It is efficient to use and it can be extended for handling physically based multi-materials.

As it follows from this review, all discussed representations for heterogeneous volumetric modelling have their advantages and drawbacks. Some of them can be used not only for defining the geometric shape of the heterogeneous object, but also for defining various attribute distributions in interior of the object. In the next chapter we will introduce a hybrid framework for heterogeneous volumetric modelling that will combine advantages of several representations and compensate for their drawbacks. This representation provides methods for defining various attributes, and the defined objects can be made time-variant.

Chapter 3

Hybrid Function Representation: Theoretical Framework

In this chapter we introduce and systematically describe a general approach for defining and generating of heterogeneous volumetric objects using a hybrid function representation (HFRep). The core of HFRep is FRep that is coupled with one of the distance function-based representations, namely, SDF, ADF or IDF. HFRep generalises their advantages and compensates for their drawbacks.

First, we give important mathematical definitions (e.g. metric space, distance function, basic set notions, function continuity, Laplace-Beltrami operator and some others) that will be further used in this and next chapters. Then we introduce formal definitions of all mentioned function-based representations. In particular, we formulate the mathematical definitions of FRep, ADF and IDF representations that to our knowledge have not been presented in the literature yet. On the basis of this, we propose a formal mathematical definition for the HFRep representation. We prove the continuity of the HFRep defining function that depends on the continuity of the FRep function and show that it preserves the Euclidean distance. We outline mathematical basics of HFRep based on FRep and SDF, FRep and ADF, and FRep and IDF hybridisations and formulate the basic algorithm for its generation. Finally, we define supported types of the objects, operations and relations.

3.1 Mathematical Background

Let us introduce the mathematical definitions which will be used hereinafter. First we introduce the definition of a metric space and a distance function that follows (H. Dyer and E. Edmunds 2014):

Definition 3.1.1. *(Metric space) Let X be a non-empty point set and let $d : X \times X \mapsto \mathbb{R}$ be such that for points $\forall \boldsymbol{p}_i \in X \subset \mathbb{R}^n$ the following conditions are satisfied:*

$$\begin{aligned} &d(\boldsymbol{p}_1, \boldsymbol{p}_2) \geq 0 \\ &d(\boldsymbol{p}_1, \boldsymbol{p}_2) = 0 \Leftrightarrow \boldsymbol{p}_1 = \boldsymbol{p}_2 \\ &d(\boldsymbol{p}_1, \boldsymbol{p}_2) = d(\boldsymbol{p}_2, \boldsymbol{p}_1) \\ &d(\boldsymbol{p}_1, \boldsymbol{p}_2) \leq d(\boldsymbol{p}_1, \boldsymbol{p}_3) + d(\boldsymbol{p}_2, \boldsymbol{p}_3) \end{aligned} \tag{3.1}$$

Then the function $d(\cdot, \cdot)$ is called a metric or a distance function on set X and the pair (X, d) is called a metric space.

Part of this chapter was published in (Tereshin, Pasko, et al. 2020, 2021).

In this work we are focusing on distance-based representations for defining volumetric objects. Let us introduce a more instrumental notion for the distance function that satisfies definition 3.1.1 and that we will use subsequently in the next sections, as follows (Schechter 1997):

Definition 3.1.2. *(Distance function) Let X be a point set in a Euclidean vector space $\mathbb{R}^n$ and let $\langle\cdot,\cdot\rangle$ be an inner product defined in $\mathbb{R}^n$. Then the Euclidean norm of the point $\boldsymbol{p} \in X$ is defined as $||\boldsymbol{p}|| = \sqrt{\langle \boldsymbol{p}, \boldsymbol{p}\rangle}$. If $\boldsymbol{q} \in X$ is another point, the distance between these two points is defined as a function:*

$$F_{DF}(\boldsymbol{p},\boldsymbol{q}) = ||\boldsymbol{p}-\boldsymbol{q}|| = \sqrt{\langle \boldsymbol{p}-\boldsymbol{q}, \boldsymbol{p}-\boldsymbol{q}\rangle}, \quad \forall \boldsymbol{p},\boldsymbol{q} \in X \subset \mathbb{R}^n \tag{3.2}$$

In this work we deal with functionally defined objects that are specified as closed point subsets $G \subseteq X$. As we are dealing with the objects defined by the functions, a point membership classification is used to distinguish between exterior, boundary and interior of the object. Therefore, let us introduce a formal definition of the boundary ∂G of the subset G as follows:

Definition 3.1.3. *(Boundary of the object) Let G be a subset of the defined metric space (X,d). The boundary ∂G of this subset G is defined as $\overline{G}\backslash G_{in}$, where $\overline{G} = \bigcap\{G_C : G_C \supseteq G\}$ is a closure of a metric space (X,d), G_C is a closed set in X, and interior of G is $G_{in} = \bigcup\{G_U : G_U \subseteq G\}$, where G_U is an open set in G.*

Both definitions of the open and closed subsets are closely related to the notion of the topology. Formally, a topology can be defined as follows (Farenick 2016; Schechter 1997):

Definition 3.1.4. *(topology) A topology on a set X is a collection of $\mathcal{T}$ of subsets X that satisfy the following three conditions:*

1. *both empty set $\emptyset$ and X are contained in $\mathcal{T}$;*
2. *$\mathcal{T}$ is closed under arbitrary unions: $\bigcup_{\alpha\in\Lambda} G_\alpha \in \mathcal{T}$, where Λ is the main set for all α;*
3. *$\mathcal{T}$ is closed under finite intersections: $\bigcap_{i=1}^{n} G_i \in \mathcal{T}$.*

The pair $(X,\mathcal{T})$ is called a topological space and its elements $G_i \in \mathcal{T}$ are called open sets. The complements of the open sets are considered as the closed sets $G_C \in \mathcal{T}$.

There are two important properties of the functions that we rely on in the next sections: continuity and smoothness. The continuity of the function is defined as follows (Delfour and Zolsio 2010):

Definition 3.1.5. *(Function continuity) Let X be an open subset of $\mathbb{R}^n$. Let $C(X)$ be the space of continuous functions $X \mapsto \mathbb{R}^n$. Let $\mathbb{N}^n$ be the set of all tuples $\alpha = (\alpha_1, ..., \alpha_n) \in \mathbb{N}^n$. Then $|\alpha|$ is the order of α and ∂^α is the partial derivative. For an integer $k \geq 1$*

$$C^k(X) := \{f \in C^{k-1}(X) : \partial^\alpha f \in C(X), \forall\alpha, |\alpha| = k\} \tag{3.3}$$

where

$$|\alpha| = \sum_{i=1}^{n} \alpha_i, \quad \partial^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} ... \partial x_N^{\alpha_N}}$$

As it was mentioned in subsection 2.3.5, distance function is a vector function and, therefore, it is Frećhete differentiable. Let us revise the formal definition of this type of derivative that is defined on a vector space. According to (Delfour and Zolsio 2010) Frećhete derivative is defined as follows:

Definition 3.1.6. *(Frećhete derivative) Let V and W be normed vector spaces and U be an open subset of V. A function $f(x) : U \mapsto W$, $x \in U$ is called Frećhete differentiable at x if $\exists A : V \mapsto W$ is a bounded linear operator that satisfies*

$$\lim_{||h|| \to 0} \frac{||f(x+h) - f(x) - Ah||_W}{||h||_V} = 0 \tag{3.4}$$

In our work we will also need a stronger continuity notion namely Lipschitz continuity. According to (Delfour and Zolsio 2010) it is defined as follows:

Definition 3.1.7. *(Lipschitz continuity) A function $f(\boldsymbol{p})$ is Lipschitz continuous on subset $X \subset \mathbb{R}^n$ if for the given $\lambda = 1$ exists $\exists c > 0$, such as $\forall \boldsymbol{p}_1, \boldsymbol{p}_2 \in X$ the following inequality holds:*

$$|f(\boldsymbol{p}_2) - f(\boldsymbol{p}_1)| \leq c|\boldsymbol{p}_2 - \boldsymbol{p}_1|^\lambda \tag{3.5}$$

where c is a Lipschitz constant.

In case of distance functions, Lipschitz constant c is equal to one. In this work we discuss functions that are either at least C^0 or C^1 continuous. A function $f(\cdot)$ is said to be of class C^0 if it is continuous on $X \subset \mathbb{R}^n$. A function $f(\cdot)$ is said to be of class C^1 if it is one time differentiable and continuous on $X \subset \mathbb{R}^n$.

Formally smoothness of the function follows from the previous definition and can be defined as (Delfour and Zolsio 2010):

Definition 3.1.8. *(Smoothness) A function $f : X \mapsto \mathbb{R}^n$ is called smooth if it is n-times differentiable, i.e. if it belongs to a specific class of functions that can be defined as $C^n(X)$ for which $f^{(n)}$ exists and it is continuous, particularly if it satisfies $C^\infty(X) = \bigcap_{n=1}^{\infty} C^n(X)$.*

The method (Rustamov, Lipman, and Funkhouser 2009) that was used for computing the interior distance fields (see section 4.3) relies on two important notions: the Laplace-Beltrami operator and its eigenfunctions. Both of them are widely used in the context of shape analysis tasks. Let us give their formal definition here.

The Laplace-Beltrami operator is a generalisation of the Laplace operator to functions defined on submanifolds in Euclidean space. According to (Delfour and Zolsio 2010) this operator can be defined as follows:

Definition 3.1.9. *(Laplace-Beltrami operator) The second order derivative of the real-valued function f defined on Euclidean space $\mathbb{R}^n$ is defined using a Laplace-Beltrami operator as*

$$\Delta_\Gamma f := \mathrm{div}_\Gamma(\nabla_\Gamma f) \tag{3.6}$$

where Γ is a submanifold of Euclidean space, Δ is a Laplacian operator, and $\mathrm{div}(\cdot)$ *is a divergence vector operator.*

Laplace-Beltrami is a linear operator. Its eigenfunctions and eigenvalues are obtained as a solution of the equation:

$$-\Delta u = \lambda u \tag{3.7}$$

where u is a non-zero eigenvector and λ is a corresponding eigenvalue.

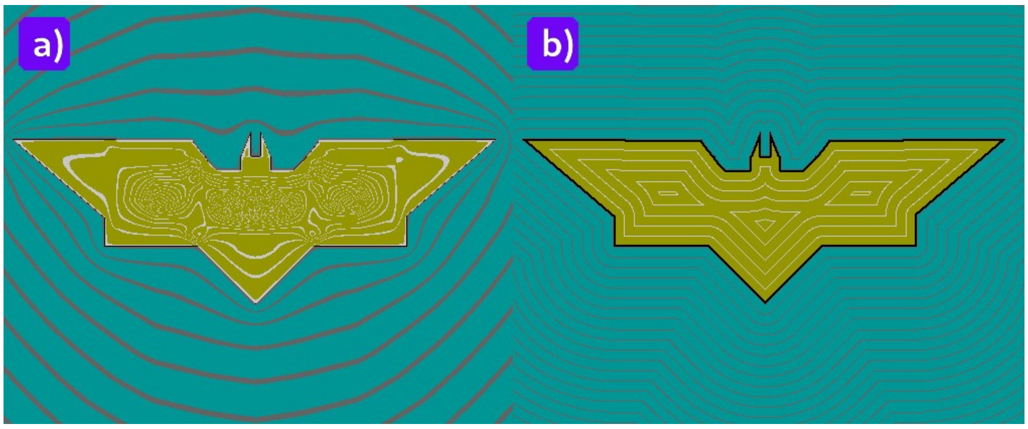


Figure 3.1: *An illustration of the FRep and SDF fields obtained for the 'bat' object defined by the FRep function. a) the FRep field; b) the SDF field.*

3.2 Formal Definitions of Function-based Representations

In this section we outline four functionally-based representations with their mathematical definitions that will be used to devise the hybrid function representation. We describe in necessary detail the mathematical basics of those representations and propose the formal definitions for three of them, namely FRep, ADF and IDF. The advantages and drawbacks of these representations were systematically outlined in chapter 2, in tables 2.4, 2.5, 2.6 and 2.7.

3.2.1 Function Representation

Let us introduce the definition of FRep (Pasko, Adzhiev, Sourin, et al. 1995):

Definition 3.2.1. *(FRep) Let the geometric shape of the object O_{FRep} be defined as a closed point subset G of n-dimensional point set X in Euclidean space $\mathbb{R}^n$ with $\boldsymbol{p} = (x_1, ..., x_n) \in \mathbb{R}^n$ using a real-valued defining function $F_{FRep}(\boldsymbol{p})$. Then function representation is defined as*

$$O_{FRep} := F_{FRep}(\boldsymbol{p}) \geq 0 \tag{3.8}$$

The FRep function provides the information about point membership:

$$\begin{cases} F_{FRep}(\boldsymbol{p}) < 0 & \boldsymbol{p} \in X \backslash G \\ F_{FRep}(\boldsymbol{p}) = 0 & \boldsymbol{p} \in \partial G \\ F_{FRep}(\boldsymbol{p}) > 0 & \boldsymbol{p} \in G_{in} \end{cases} \tag{3.9}$$

where X is a subset of Euclidean space $\mathbb{R}^n$, $X \backslash G$ is an exterior of the geometric shape G, ∂G is the boundary of the geometric shape G of the object O_{FRep} and G_{in} is the interior of the G. The major requirement for $F_{FRep}(\boldsymbol{p})$ is to be at least C^0 continuous. Note, that in this work we will consistently follow this point-membership convention for all types of distance functions, meaning that they are positive inside and negative outside with respect to the zero-level set.

The FRep function (see Fig. 3.1, a) provides the information about point membership, system of equations (3.9). The major requirement for $F_{FRep}(\boldsymbol{p})$ is to be at

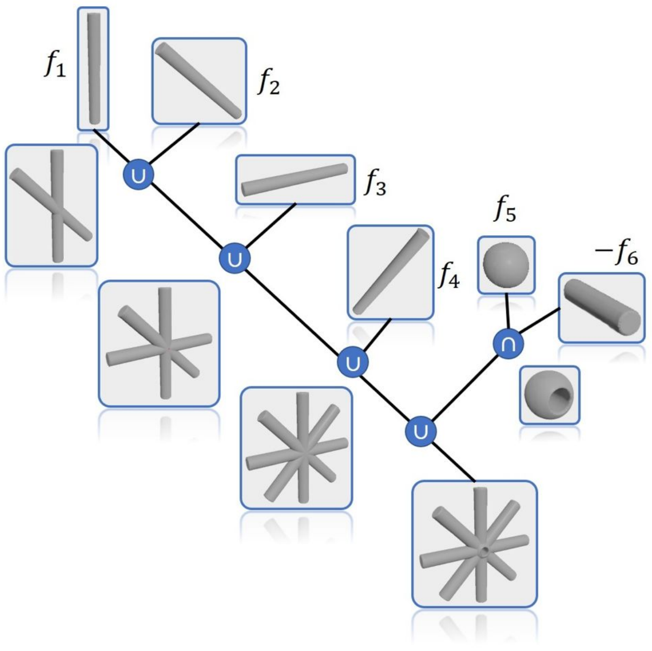


Figure 3.2: *A constructive tree for the FRep object in the form of a 'snow flake' that was converted to SDF. This tree consists of objects defined by SDF functions f_i stored in the tree leafs and operations applied to them stored in the tree nodes.*

least C^0 continuous. The main advantages and drawbacks of FRep were discussed in subsection 2.3.3 and are presented in table 2.4.

FRep is a high-level and uniform representation of multidimensional geometric objects. The subject of particular interest is 4D objects with fourth coordinate specified as time. FRep generalises implicit surface modelling and extends a CSG approach. FRep has a closure property as operations applied to the FRep defining functions produce continuous resulting FRep functions. The FRep object can be defined as a primitive (e.g. sphere, torus, cylinder, octahedron, etc.) or as a complex object that is defined in the form of a constructive tree. In this case, primitives are stored in the leaves of the tree and operations are stored in its nodes.

Fig. 3.1 (a) shows the FRep field obtained using 14 set-theoretic operations applied to triangles and rectangles to construct the 'bat'. In Fig. 3.2 we present a constructive tree that describes how a FRep object in the form of a 'snow flake', that was converted to SDF, was created using union $\cup$ and intersection $\cap$ set-theoretic operations. In general case, the FRep field is not distance-based as interior and exterior field isolines do not precisely follow the object shape in terms of its zero-level set boundary.

3.2.2 Signed Distance Function

Let us introduce the definition of SDF that relies on definitions 3.1.1, 3.1.2 and 3.1.3:

Definition 3.2.2. *(SDF) Let (X, d) be a metric space. Let the geometric shape G of the object O_{SDF} be specified in (X, d) as a point subset $G \subseteq X$. Then a signed distance function $F_{SDF}(\boldsymbol{p})$ is defined as:*

$$F_{SDF}(\boldsymbol{p}) = \begin{cases} d(\boldsymbol{p}, \partial G) & \textit{if} \quad \boldsymbol{p} \in G \\ -d(\boldsymbol{p}, \partial G) & \textit{otherwise} \end{cases} \tag{3.10}$$

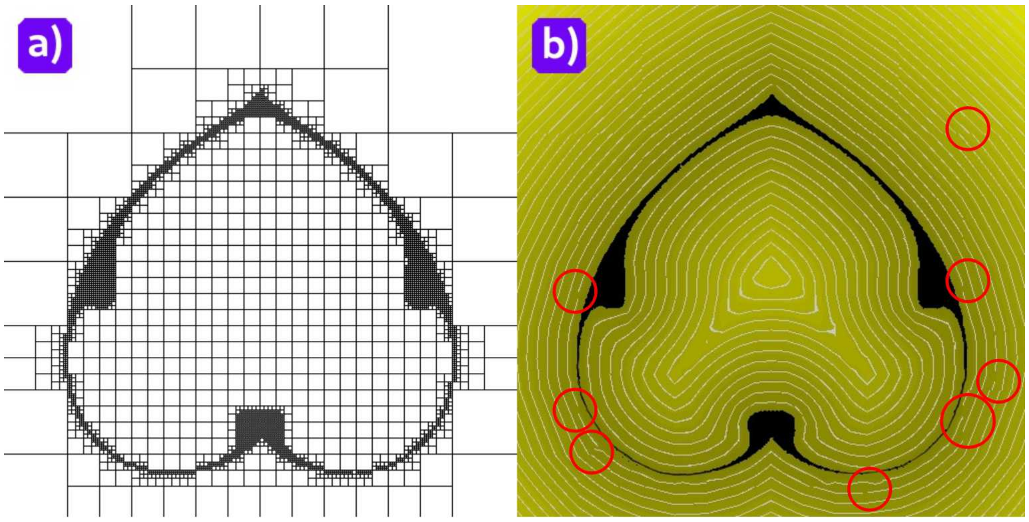


Figure 3.3: *An ADF field generated for the FRep object 'heart'. The distance field is restored using bilinear interpolation. a) The constructed quadtree for the 'heart' object. b) The computed unsigned distance field with some C^0 discontinuities shown in the red circles.*

where $d(\boldsymbol{p}, \partial G) \equiv F_{DF}(\boldsymbol{p}, \partial G)$. Then the SDF representation is defined as follows:

$$O_{SDF} := F_{SDF}(\boldsymbol{p}) \geq 0 \tag{3.11}$$

SDF is at least C^0 continuous as it might be not differentiable at some points of Euclidean space $\mathbb{R}^n$ and it has gradient discontinuities on the object's medial axes. This means that SDF has the property of Lipschitz uniform continuity. The SDF representation conventionally provides the information about point membership in the same manner as FRep. SDF satisfies the solution of the eikonal equation. This means that the Euclidean distance is preserved. The main advantages and drawbacks of SDF were discussed in subsection 2.3.5 and are presented in table 2.5.

Fig. 3.1 (b) shows the SDF field generated for the 'bat' object. As it can be seen, the isolines are spaced equidistantly and follow the shape of the object.

3.2.3 Adaptively Sampled Distance Function

Adaptively sampled distance function (ADF) (Frisken et al. 2000) is a distance function that is computed on hierarchical grids, e.g. tree-like data structures. ADF satisfies all the requirements of definitions 3.1.1, 3.1.2 and 3.2.2. To our knowledge, there is no well-established formal definition of ADF in the literature. In this work we introduce a formal definition of ADF.

Let us first give the definition of the hierarchical tree structure:

Definition 3.2.3. *(Hierarchical tree structure) Let a set of nodes and edges (Q, E) be an undirected connected graph T that contains no loops and starts at some particular node of T. Then such a graph T is defined as a tree.*

Let the space $\mathbb{R}^n$ be subdivided according to the local details using some k-ary tree $T := (Q, E)$ with nodes $q \in Q$. Each node q is defined as an n-dimensional cell. According to the SDF definition 3.2.2 we need to compute the distance to the boundary ∂G of the geometric subset G. Taking these preliminaries into account, let us give the ADF definition:

Definition 3.2.4. *(ADF) Let the geometric shape $G \subseteq X$ of the object O_{ADF} be defined in a metric space (X, d). Let (X, d) be subdivided into nodes $q \in Q$ with corner vertices $\boldsymbol{p}_i$ according to the level of detail using k-ary tree $T := (Q, E)$. Let the boundary ∂G be subdivided with the maximum tree depth, while $X \backslash G$ and G_{in} be subdivided with some minimum tree depth. Let the corner vertices of the boundary nodes q be defined as $\boldsymbol{p}_{b_i}$. Then the distance function between these points is $d(\boldsymbol{p}_i, \boldsymbol{p}_{b_i}) \equiv F_{DF}(\boldsymbol{p}_i, \boldsymbol{p}_{b_i})$. Thereafter, the ADF distance function $F_{ADF}(\boldsymbol{p})$ on the tree T is restored at each node q using some interpolation function $F_I(\boldsymbol{p})$ and is defined as follows:*

$$F_{ADF}(\boldsymbol{p}) = \begin{cases} (F_I \circ F_{DF})(\boldsymbol{p}) & \text{if} \quad \boldsymbol{p} \in G \\ -(F_I \circ F_{DF})(\boldsymbol{p}) & \text{otherwise} \end{cases} \tag{3.12}$$

The ADF representation is defined in the form of an inequation:

$$O_{ADF} := F_{ADF}(\boldsymbol{p}) \geq 0 \tag{3.13}$$

The ADF field has C^0 discontinuities on the edges of two adjacent cells of different size and it also has C^1 discontinuities caused by the bilinear/trilinear interpolation that was used for restoring DF at each cell (Frisken et al. 2000). The ADF representation provides the information about point membership in the same conventional manner as FRep. The generated ADF field for the FRep 'heart' object can be seen in Fig. 3.3. The distance field was restored using bilinear interpolation, therefore, the field is non-smooth and has C^0 discontinuities (some isolines are disconnected). The advantages and drawbacks of the ADF representation were discussed in subsection 2.3.6 and given in table 2.6.

3.2.4 Interior Distance Function

Interior distance function (IDF) is not a well-established notion yet as in literature there is neither a general approach for generating DFs of this rather broad nature nor one unique name for them. Different approaches for IDF definition were discussed in the previous chapter in subsection 2.3.7 as well as its advantages and drawbacks in table 2.7.

In this work we suggest to use this notion for a representation with a defining function obtained as follows: the distance function is computed on the boundary of the object as a shortest path between boundary points and then the generated distances are smoothly interpolated in the object's interior. Let us introduce the definition of IDF:

Definition 3.2.5. *(IDF) Let the geometric shape $G \subseteq X$ of the object O_{IDF} be defined in a metric space (X, d). Let points $\boldsymbol{p}_{b_i}$ belong to ∂G, and let points $\boldsymbol{p}_{in_k}$ belong to G_{in}. Let a distance function $d(\boldsymbol{p}_{b_i}, \boldsymbol{p}_{b_j}) \equiv F_{DF}(\boldsymbol{p}_{b_i}, \boldsymbol{p}_{b_j}) = ||\boldsymbol{p}_{b_i} - \boldsymbol{p}_{b_j}||_{\mathbb{R}^n}$ between any boundary points $\boldsymbol{p}_{b_i}$ and $\boldsymbol{p}_{b_j}$ on a curved domain ∂G be recovered. Thereafter, by constructing an interpolation function $F_I(F_{DF}(\boldsymbol{p}_{b_i}, \boldsymbol{p}_{b_j}), \boldsymbol{p}_{in_k})$ that is at least C^1 continuous, boundary distances are extended to interior of the object O_{IDF}. Therefore, the IDF function can be defined as:*

$$F_{IDF}(\boldsymbol{p}_{in_k}) = F_I(F_{DF}(\boldsymbol{p}_{\boldsymbol{b_i}}, \boldsymbol{p}_{\boldsymbol{b_j}}), \boldsymbol{p}_{in_k}) \tag{3.14}$$

where $0 \leq i, j, < N$, N is the number of boundary points, $0 \leq k < M$, M is the number of interior points. The IDF representation is defined in the form of an inequation:

$$O_{IDF} := F_{IDF}(\boldsymbol{p}) \geq 0 \tag{3.15}$$

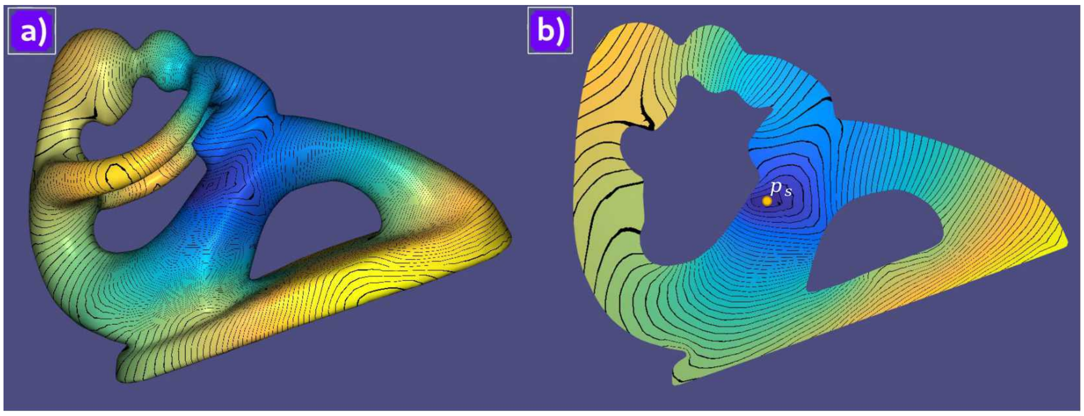


Figure 3.4: *An IDF field generated for the BRep tetrahedralised object 'fertility statue' using method presented in (Rustamov, Lipman, and Funkhouser 2009). a) The IDF field computed on the boundary of the object. b) The slice of the IDF field computed in interior of the object. The distances are computed between 'source' point $\boldsymbol{p}_s$ and the other points of the tetrahedral mesh.*

Here the interpolation function $F_I(\cdot)$ will depend on a particular representation of the object. In case when the volumetric object is defined using voxel grids, spline-based interpolations can be used. If the volumetric object is defined as a tetrahedralised mesh (i.e. formed out of tetrahedras), then data can be interpolated using barycentric interpolation. Note, that the boundary of the object is treated as the source for generating interior distances.

In Fig. 3.4 we show an example of the computed IDF field for the tetrahedral object of the 'fertility statue'. A simple procedure of IDF isolines generation can be considered as a computation of distances from the fixed 'source' point $\boldsymbol{p}_s$ to other points of the mesh. The obtained IDF field at the boundary and in the interior of the mesh is smooth and shape-aware. The IDF field was generated using the method proposed in (Rustamov 2011) and it is at least C^1 continuous.

3.2.5 Hypervolume Representation

In this work we extend one of the frameworks discussed in section 2.4 for defining heterogeneous objects. This framework was introduced in (Pasko, Adzhiev, Schmitt, et al. 2001) and called the hypervolume representation. A hypervolume object is defined as follows:

Definition 3.2.6. *(Hypervolume) Let the geometric shape G of O_{H_V} be defined by a real-valued function $F_G(\boldsymbol{p}), \boldsymbol{p} \in \mathbb{R}^n$ that is at least C^0 continuous and let attributes be defined by any $F_{A_i}(\boldsymbol{p})$. Then heterogeneous object O_{H_V} is defined as:*

$$O_{H_V} := (G, A_1, ..., A_n) : (F_G(\boldsymbol{p}), F_{A_1}(\boldsymbol{p}), ..., F_{A_n}(\boldsymbol{p})), \tag{3.16}$$

where $n \in \mathbb{N}$ is the number of attributes.

In general case, attribute functions $F_{A_i}(\boldsymbol{p})$ are not necessarily continuous. However, as it was shown in (Biswas, Shapiro, and Tsukanov 2004), better control of the attributes definition on the surface and in the interior of the distance-based objects can be achieved when the attribute defining functions are parameterised by the distances. This concept will be further discussed in this chapter.

3.3 Hybrid Function Representation (HFRep): Requirements

Let us formulate the requirements for the heterogeneous hybrid framework. Our goal is to introduce the formalised and well established theoretical and practical framework for heterogeneous object modelling that includes substantiated theory, methods for modelling, attribute definition and operations to deal with supported objects. The core of this framework is a proposed hybrid function representation (HFRep) that is suitable for defining volumetric heterogeneous objects. We assume that the geometric shape G of the given object is defined by FRep, and its defining function is known. To devise the HFRep embracing advantages and circumventing disadvantages of FRep, SDF, ADF, IDF, it is essential to obtain a real-valued defining function in an n-dimensional Euclidean space with the following properties:

1. The HFRep function should provide sufficiently accurate distance approximation in Euclidean space $\mathbb{R}^n$.
2. The HFRep function should be at least C^0 continuous with possibility to enforce it to be at least C^1 continuous.
3. The HFRep function should satisfy the point membership test: it should be positive in interior of the geometric shape G, take exact zero values only at the object boundary ∂G and it should be negative in exterior of the geometric shape $X\backslash G$;
4. The HFRep should be a multidimensional object representation; in particular, dealing with 4D objects is of paramount importance to cover time-variant models with the fourth 'time' coordinate;
5. The HFRep representation should be suitable for the heterogeneous object modelling allowing for defining attribute functions related to the geometry;
6. The HFRep attribute functions should depend on evaluation point $\boldsymbol{p} \in G$ and be parameterised by distance values of the obtained HFRep geometry function.

The defining function of the HFRep object will be smooth (see definition 3.1.8) if it is at least one times differentiable. Overall, the defining HFRep function that is considered in concert with attribute functions parameterised by distances will be suitable for dealing with multi-material aspects of heterogeneous objects including time-variant ones.

3.4 HFRep: Definition

First, we provide a mathematical definition for the geometric aspects of HFRep. Then we add the part related to attributes. The geometric shape G of an HFRep object O_{HFRep} is defined by geometric function $F_G(\boldsymbol{p})$ as follows:

Definition 3.4.1. *(HFRep, geometric shape) Let the geometric shape $G \subseteq X$ of the object O_{HFRep} be defined in a metric space (X, d). Given at least C^0 or C^1 continuous FRep function $F_{FRep}(\boldsymbol{p})$, the distance to the object boundary ∂G is defined as $(F_I \circ F_{DF})(\boldsymbol{p}, \partial G) \equiv (F_I \circ F_{DF})(\boldsymbol{p})$, where $F_I(\cdot)$ is at least C^1 continuous interpolation*

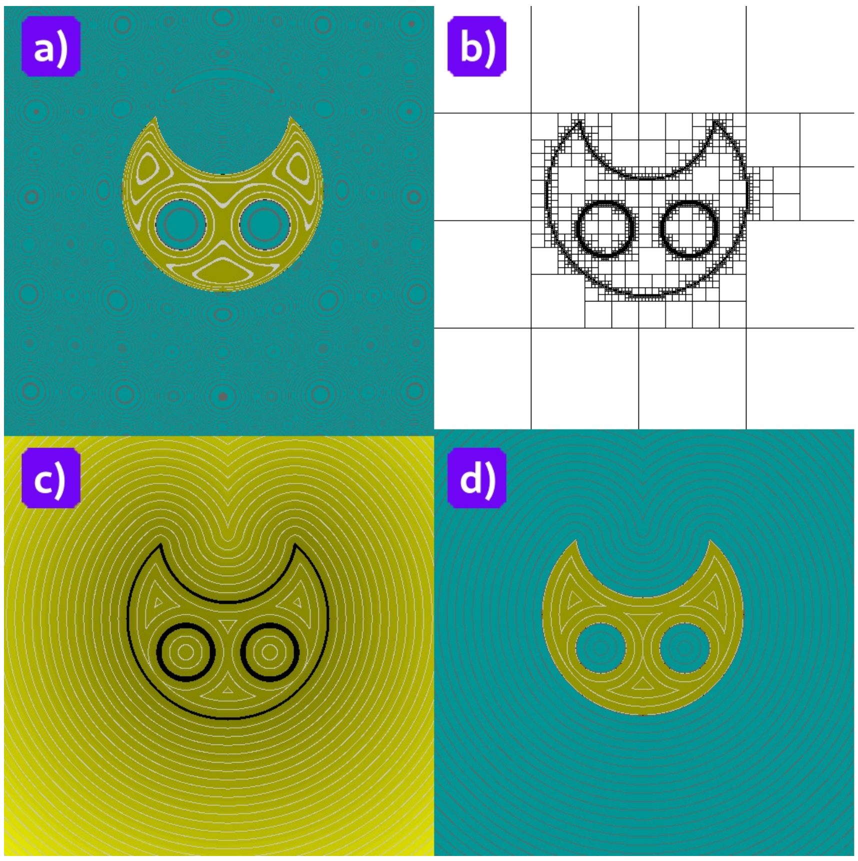


Figure 3.5: *The illustration of HFRep based on FRep and ADF with applied PHT-spline (a polynomial spline over hierarchical T-mesh) interpolation to restore the distance field at each cell. ADFs are generated using a numerical solution of the eikonal equation on the quadtree. a) the FRep field; b) a hierarchical quadtree subdivision; c) UDF computed on the quadtree with the applied PHT-spline interpolation for restoring distances at each quadtree cell; d) the HFRep field that was obtained using the generated ADF.*

function and $d(\cdot,\cdot) \equiv F_{DF}(\cdot,\cdot)$ *is an unsigned distance-based function, in particular SDF, ADF or IDF. Then the HFRep function is defined as follows:*

$$F_G(\boldsymbol{p}) := F_{HFRep}(\boldsymbol{p}) = (F_{sign} \circ F_{FRep})(\boldsymbol{p}) \cdot (F_I \circ F_{DF})(\boldsymbol{p}) \tag{3.17}$$

where $F_{sign}(\cdot)$ *is an at least* C^1 *continuous function that provides a sign for the computed function* $(F_I \circ F_{DF})(\boldsymbol{p})$ *and satisfies the FRep point membership test, system of inequations (3.9). Finally, the HFRep representation is defined as:*

$$O_{HFRep} := F_{HFRep}(\boldsymbol{p}) \geq 0 \tag{3.18}$$

We suggest to consider spline-based interpolation functions $F_I(\cdot)$ as they provide a smooth approximation of the computed discrete unsigned distance field. For more details refer to section 4.4.

The continuity of the HFRep function $F_{HFRep}(\boldsymbol{p})$ depends on the continuity of the FRep function $F_{FRep}(\boldsymbol{p})$. In the case when we are dealing only with geometric shapes, it is sufficient to have C^0 continuity for the HFRep function. Otherwise, in case of heterogeneous object modelling, the HFRep function should belong to the class of functions that are at least C^1 continuous. This means that the HFRep function have to be Lipschitz continuous, i.e. C^0 continuous. This property is inherited from the SDF representation.

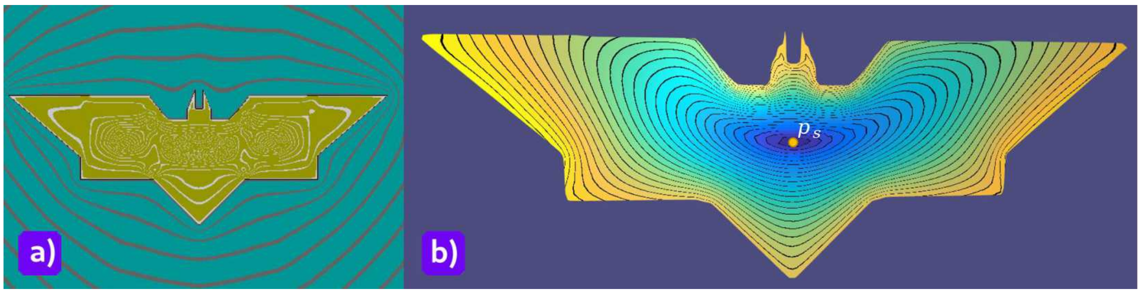


Figure 3.6: *HFRep based on FRep and IDF. (a) 'Bat' object and its FRep field; (b) the HFRep 'bat' object generated on the basis of the FRep object. The isolines and colour show how the field changes from the source point $\boldsymbol{p}_s$ towards the object boundary.*

Inherited from FRep	Inherited from SDF	Inherited from ADF	Inherited from IDF
• The continuity of the HFRep function depends on the continuity of the FRep function. • HFRep represents multidimensional objects, in particular 4D objects with the fourth coordinate specified as time.	• HFRep provides at least C^0 continuous distance function. • The HFRep function is Lipschitz continuous; • The HFRep function satisfies the solution of the eikonal equation; • The HFRep object can be efficiently discretised and rendered.	• HFRep provides at least C^0 continuous distance function for any FRep object that was spatially subdivided according to the local details using a hierarchical data structure. • Hierarchical data structure can also be used for defining and storing object's attributes.	• HFRep provides at least a C^0 continuous unsigned distance function for any FRep object in its interior if IDF is used for obtaining distances; • Distances in the interior of the HFRep object are shape-aware and deformed with boundaries. • There is also a potential for modelling attributes in interior of the volumetric object.

Table 3.1: *Properties of the hybrid function representation.*

The HFRep function inherits the SDF function property, which preserves Euclidean distance that can be computed as a solution of the eikonal equation. In the next subsection we will formally prove that the HFRep function has these properties.

Now on the basis of definition 3.2.6, we can formulate the definition of the heterogeneous HFRep object $O_{H_{V,HFRep}}$ as follows:

Definition 3.4.2. *(HFRep, heterogeneous object) Let the geometric shape G of $O_{H_{V,HFRep}}$ be defined by at least C^1 continuous $F_G(\boldsymbol{p}) = F_{HFRep}(\boldsymbol{p})$ distance-based function. Let the attribute A_i be defined as a real-valued function $F_{A_i}(F_{HFRep}(\boldsymbol{p}), \boldsymbol{p})$. Then the HFRep heterogeneous object $O_{H_{V,HFRep}}$ is defined as:*

$$O_{H_{V,HFRep}} := \begin{cases} F_G(\boldsymbol{p}) := F_{HFRep} \geq 0 \\ F_{A_i}(F_{HFRep}(\boldsymbol{p}), \boldsymbol{p}), \quad i = [0, .., n] \in \mathbb{N} \end{cases} \tag{3.19}$$

where n is a number of attributes.

In table 3.1 we outline properties that were inherited by HFRep from FRep, SDF, ADF and IDF representations.

In Fig. 3.5 (d), we demonstrate the restored distance field computed on the hierarchical grid obtained for the initial FRep object defined as a half circle with two holes, Fig. 3.5 (a). There is neither C^0 nor C^1 discontinuities in the field as it can be seen in Fig. 3.5 (c) or (d). All the isolines are smooth and continuous.

In Fig. 3.6 (b) we show a simple example of interior distances computed for the FRep 'bat' object, Fig. 3.6 (a), that was constructed using 14 set-theoretic operations. First, the boundary of the FRep object was extracted for computing

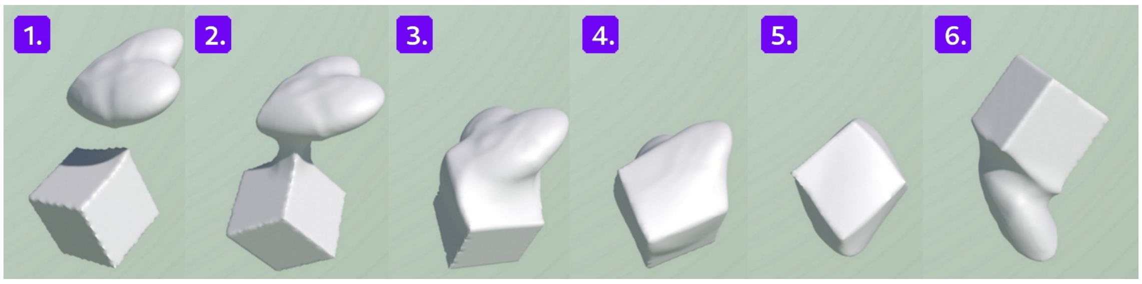


Figure 3.7: *The STB-based metamorphosis operation over the initially FRep 'heart' converted to HFRep and initially BRep 'cube' converted to SDF 'cube'. Supplementary video: figure3.7.mpg.*

boundary distances. Then, the interior of the obtained convex contour was triangulated. Finally, the boundary distances were propagated in interior of the shape as it is described in (Rustamov, Lipman, and Funkhouser 2009). The black isolines show that the obtained field is at least C^1 continuous as they are smoothly changing in the object interior.

Fig. 3.7 shows a metamorphosis between two oscillating 4D geometric shapes ('heart', initially the FRep object then converted to HFrep; 'cube', initially the BRep object then converted to SDF) using the space-time blending (STB) method (Pasko, Pasko, and Kunii 2005; Tereshin, Anderson, et al. 2020). The result of application of this method is a non-distance based continuous function that defines an object.

3.5 The Hybrid Function Representation: Mathematical Properties

In this section we will formally prove that the HFRep function has the following properties:

- the continuity of the HFRep function depends on the continuity of the FRep function and it is at least C^0 or C^1;
- the gradient of the HFRep function satisfies the eikonal equation;

Let us formally prove that the continuity of the HFRep function defined by equation (3.17) depends on the continuity of the FRep function. To do this, first, we will prove that the composition of a function for the sign definition $F_{sign}(\cdot)$ and a FRep function $F_{FRep}(\boldsymbol{p})$ is at least C^0 continuous.

Proposition 3.5.1. *Let $F_{sign}(\cdot)$ be a function that $F_{sign}(\cdot) : \mathbb{R} \mapsto \mathbb{R}$ and $F_{sign}(\cdot) \in C^1(\mathbb{R}, \mathbb{R})$. Let the FRep function $F_{FRep}(\boldsymbol{p})$ be a function that $F_{FRep}(\boldsymbol{p}) : \mathbb{R}^n \mapsto \mathbb{R}$ that is either $F_{FRep}(\boldsymbol{p}) \in C^0(\mathbb{R}^n, \mathbb{R})$ or $F_{FRep}(\boldsymbol{p}) \in C^1(\mathbb{R}^n, \mathbb{R})$. Then the continuity of the composition of these two functions $(F_{sign} \circ F_{FRep})(\boldsymbol{p})$ depends on the continuity of the FRep function $F_{FRep}(\boldsymbol{p})$ and is either $C^0(\mathbb{R}, \mathbb{R})$ or $C^1(\mathbb{R}, \mathbb{R})$.*

Proof. Let the function $F_{sign}(\cdot)$ be defined on some set $B \subset \mathbb{R}$ and the FRep function $F_{FRep}(\cdot)$ be defined on some set $A \subset \mathbb{R}$. Let us assume that $F_{FRep}(A) \subset B$. If $F_{FRep}(\boldsymbol{p})$ is continuous at some point $\boldsymbol{p} \in A$ and $F_{sign}(F_{FRep}(\boldsymbol{p}))$ is continuous at $F_{FRep}(\boldsymbol{p}) \in B$, then $(F_{sign} \circ F_{FRep})(\cdot) \in A$ and is continuous at $\boldsymbol{p}$. The proof of this assumption will show that this composition is at least $C^0(\mathbb{R}, \mathbb{R})$.

Given some $\epsilon > 0$, as it was stated, $F_{sign}(\cdot)$ is continuous at $F_{FRep}(\boldsymbol{p})$, then:

$$\exists \eta > 0 : |\boldsymbol{p}_1 - F_{FRep}(\boldsymbol{p})| < \eta, \boldsymbol{p}_1 \in B \Leftrightarrow |F_{sign}(\boldsymbol{p}_1) - F_{sign}(F_{FRep}(\boldsymbol{p}))| < \epsilon \tag{3.20}$$

As $F_{FRep}(\cdot)$ is continuous in $\boldsymbol{p} \in A$, then:

$$\exists \delta : |\boldsymbol{p}_2 - \boldsymbol{p}| < \delta, \boldsymbol{p}_2 \in A \Leftrightarrow |F_{FRep}(\boldsymbol{p}_2) - F_{FRep}(\boldsymbol{p})| < \eta \tag{3.21}$$

Taking into account inequations (3.20) and (3.21) we get that:

$$|\boldsymbol{p}_2 - \boldsymbol{p}| < \delta, \boldsymbol{p}_2 \in A \Leftrightarrow |F_{sign}(F_{FRep}(\boldsymbol{p}_2)) - F_{sign}(F_{FRep}(\boldsymbol{p}))| < \epsilon \tag{3.22}$$

This inequation implies that $(F_{sign} \circ F_{FRep})(\cdot)$ is at least $C^0(\mathbb{R}, \mathbb{R})$ continuous.

To show that $(F_{sign} \circ F_{FRep})(\cdot)$ is at least $C^1(\mathbb{R}, \mathbb{R})$ continuous, we have to prove that at least first derivative of this function composition $F'_{sign}(F_{FRep}(\boldsymbol{p}))F_{FRep}(\boldsymbol{p})'$ exists.

$F'_{sign}(F_{FRep}(\boldsymbol{p}))$ is differentiable at $F_{FRep}(\boldsymbol{p})$ and $F_{FRep}(\boldsymbol{p})'$ is differentiable at $\boldsymbol{p}$ only if the following limits exist:

$$\begin{aligned} F_{sign}(F_{FRep}(\boldsymbol{p}))' &= \lim_{|\boldsymbol{h}| \longrightarrow 0} \frac{F_{sign}(F_{FRep}(\boldsymbol{p}+\boldsymbol{h})) - F_{sign}(F_{FRep}(\boldsymbol{p}))}{|\boldsymbol{h}|} \\ F_{FRep}(\boldsymbol{p})' &= \lim_{|\boldsymbol{h}| \longrightarrow 0} \frac{F_{FRep}(\boldsymbol{p}+\boldsymbol{h}) - F_{FRep}(\boldsymbol{p})}{|\boldsymbol{h}|} \end{aligned} \tag{3.23}$$

To show that this statement is true, let us assume that $F'_{sign}(F_{FRep}(\cdot))$ is differentiable at $F_{FRep}(\boldsymbol{p})$ and $F_{FRep}(\boldsymbol{p})'$ is differentiable at $\boldsymbol{p}$. Then we can write the following:

$$\begin{aligned} F_{sign}(F_{FRep}(\boldsymbol{p}))' &= \lim_{\boldsymbol{x} \longrightarrow \boldsymbol{p}} \frac{F_{sign}(F_{FRep}(\boldsymbol{x})) - F_{sign}(F_{FRep}(\boldsymbol{p}))}{|\boldsymbol{x}-\boldsymbol{p}|} \\ F_{FRep}(\boldsymbol{p})' &= \lim_{\boldsymbol{x} \longrightarrow \boldsymbol{p}} \frac{F_{FRep}(\boldsymbol{x}) - F_{FRep}(\boldsymbol{p})}{|\boldsymbol{x}-\boldsymbol{p}|} \end{aligned} \tag{3.24}$$

Let $\boldsymbol{h} = \boldsymbol{x} - \boldsymbol{p}$ and as $\boldsymbol{x} \rightarrow \boldsymbol{p}$ then $|\boldsymbol{h}| \rightarrow 0$ and two equalities (3.23) holds. This means that if $F_{sign}(F_{FRep}(\cdot))$ and $F_{FRep}(\cdot)$ are differentiable, then their limits must exist and vice versa. In this case the composition $(F_{sign} \circ F_{FRep})(\boldsymbol{p})$ is at least $C^1(\mathbb{R}, \mathbb{R})$ continuous. □

Now, let us prove that the composition of the smoothing function $F_I(\cdot)$ and unsigned distance function $F_{DF}(\boldsymbol{p})$ is at least C^1 continuous.

Proposition 3.5.2. *Let $F_I(\cdot)$ be an interpolation function that $F_I(\cdot) : \mathbb{R} \mapsto \mathbb{R}$ and that is $F_I(\cdot) \in C^1(\mathbb{R}, \mathbb{R})$. Let $F_{DF}(\boldsymbol{p})$ be a distance function that $F_{DF}(\boldsymbol{p}) : \mathbb{R}^n \mapsto \mathbb{R}$ and that is $F_{DF}(\boldsymbol{p}) \in C^0(\mathbb{R}^n, \mathbb{R})$. Then the composition of these functions $(F_I \circ F_{DF})(\boldsymbol{p})$ is at least $C^1(\mathbb{R}, \mathbb{R})$ continuous.*

Proof. Let us assume that we have three metric spaces (X_p, d_p), (X, d) and (X_{sm}, d_{sm}). $X_p \subset \mathbb{R}^n$ is a set of points with a distance metric $d_p(\boldsymbol{p}_1, \boldsymbol{p}_2)$, $X \subseteq X_p \subset \mathbb{R}^n$ is a set of values of the $F_{DF}(\cdot)$ function in points of the set X_p with distance metric $d(F_{DF}(\boldsymbol{p}_1, F_{DF}(\boldsymbol{p}_2)))$ and $X_{sm} \subseteq X \subseteq X_p \subset \mathbb{R}^n$ is a set of the smoothed values of $F_{DF}(\cdot)$ with a distance metric $d_{sm}((F_I \circ F_{DF})(\boldsymbol{p}_1), (F_I \circ F_{DF})(\boldsymbol{p}_2))$.

First, we show that the composition $(F_I \circ F_{DF})(\cdot) : X_p \mapsto X_{sm}$ is uniformly continuous on the set X_p that is equivalent $(F_I \circ F_{DF})(\cdot) \in C^0(\mathbb{R}, \mathbb{R})$.

Let us assume that $F_{DF}(\cdot)$ is uniformly continuous on X_p and $F_I(\cdot)$ is uniformly continuous on $F_{DF}(X_p)$. Given $\epsilon > 0$, we want to find $\delta > 0 : \{\forall \boldsymbol{p}, \boldsymbol{q} \in X_p, d_p(\boldsymbol{p}, \boldsymbol{q}) < \delta\}$ such that:

$$d_{sm}(F_I(F_{DF}(\boldsymbol{p})), F_I(F_{DF}(\boldsymbol{q}))) < \epsilon \tag{3.25}$$

A function $F_I(\cdot)$ is uniformly continuous on a set X if $\exists \delta_1 > 0 : \{F_{DF}(\boldsymbol{p}), F_{DF}(\boldsymbol{q}) \in X, d(F_{DF}(\boldsymbol{p}), F_{DF}(\boldsymbol{q})) < \delta_1\}$ and inequation (3.25) hold.

A function $F_{DF}(\cdot)$ is uniformly continuous on a set X_p if $\exists \delta_2 > 0 : \{\boldsymbol{p}, \boldsymbol{q} \in X_p, d_p(\boldsymbol{p}, \boldsymbol{q}) < \delta_2\}$ and the following inequation hold:

$$d(F_{DF}(\boldsymbol{p}), F_{DF}(\boldsymbol{q})) < \delta_1 \tag{3.26}$$

Let $\delta = \delta_2$. Then $\forall \boldsymbol{p}, \boldsymbol{q} \in X_p$ inequation (3.26) holds true and subsequently inequation (3.25) also holds true. Thus $(F_I \circ F_{DF})(\cdot) : \{X_p \mapsto X_{sm}\} \in C^0(\mathbb{R}, \mathbb{R})$ is uniformly continuous on X_p.

To show that the composition $(F_I \circ F_{DF})(\boldsymbol{p})$ is $C^1(\mathbb{R}^n, \mathbb{R})$ we have to prove that $F_{sm}(\boldsymbol{p})' = ((F_I \circ F_{DF})(\boldsymbol{p}))'$ exists in all points of set $X_p \subset \mathbb{R}^n$. The C^1 continuity of the function $F_{DF}(\boldsymbol{p})$ can be provided only by the smoothing interpolation function $F_I(\cdot)$ that is at least C^1 continuous and sufficiently accurate approximates the values of $F_{DF}(\boldsymbol{p})$. This interpolation function $F_I(\cdot)$ should satisfy the so called K-functional of Peetre $K_{r,p}(t)$ (Schumaker 2007) that can be defined for the function $F_{DF}(\boldsymbol{p})$ as follows:

$$K_{r,p}(t)F_{DF} = \inf_{g \in L_p^r[I]} (||F_{DF} - (F_I \circ F_{DF})||_p + t^r ||D^r(F_I \circ F_{DF})||_p) \tag{3.27}$$

where $1 \leq p < \infty$, $t > 0$ and $r > 0$ is an integer, $L_p[I]$ is a Lebesgue space defined on a space of bounded real-valued functions I in a closed finite interval $[a, b]$. Here D^r is a differential operator that defines the r^{th} derivative, and t controls the balance between the size of the derivative and approximation error. The K-functional defines how well the function $F_{DF}(\boldsymbol{p})$ can be approximated by smooth functions. It means, that in our case, the interpolation function $F_I(\cdot)$, which will approximate the distance function $F_{DF}(\boldsymbol{p})$, should be at least $r = 1$ times differentiable, while the value of the K-functional $K_{r,p}(t)$ should stay minimum. $\square$

Therefore, the continuity of the HFRep function is defined as:

$$C_{HFRep} = \min(C^m_{(F_{sign} \circ F_{FRep})(\boldsymbol{p})}, C^k_{(F_I \circ F_{DF})(\boldsymbol{p})}) \tag{3.28}$$

where $m = 0$ or $m = 1$, $k = 1$, i.e. the minimum class of continuity between two function compositions.

Let us formally prove that the gradient of the HFRep function satisfies the eikonal equation. This property is essential to show that the Euclidean distance is preserved by the HFRep function.

Proposition 3.5.3. *Let the HFRep function $F_{HFRep}(\boldsymbol{p}) : \mathbb{R}^n \mapsto \mathbb{R}$ be given. Then we state that the following holds true:*

$$||\nabla F_{HFRep}(\boldsymbol{p})|| \approx 1 \tag{3.29}$$

Proof. Let us recall the HFRep function:

$$F_{HFRep}(\boldsymbol{p}) = (F_{sign} \circ F_{FRep})(\boldsymbol{p}) \cdot (F_I \circ F_{DF})(\boldsymbol{p})$$

Now we can substitute it to equation (3.29) and rewrite it as follows:

$$\left(\frac{\partial((F_{sign} \circ F_{FRep})(p_1) \cdot (F_I \circ F_{DF})(p_1)}{\partial p_1}\right)^2 + ... + \left(\frac{\partial((F_{sign} \circ F_{FRep})(p_n) \cdot (F_I \circ F_{DF})(p_n))}{\partial p_n}\right)^2 \approx 1 \tag{3.30}$$

Here we take into account that HFRep function is a real valued function. In this case we can omit the square root. For simplicity, we assume that $(F_{sign} \circ F_{FRep})(p_i) \equiv F_{sign} \circ F_{FRep}$ and $(F_I \circ F_{DF})(p_i) \equiv F_I \circ F_{DF}$. Then we can open the brackets and combine the terms accordingly:

$$\begin{aligned}&\left((F_I \circ F_{DF})\frac{\partial(F_{sign} \circ F_{FRep})}{\partial p_1} + (F_{sign} \circ F_{FRep})\frac{\partial(F_I \circ F_{DF})}{\partial p_1}\right)^2 + ... \\ &\left((F_I \circ F_{DF})\frac{\partial(F_{sign} \circ F_{FRep})}{\partial p_n} + (F_{sign} \circ F_{FRep})\frac{\partial(F_I \circ F_{DF})}{\partial p_n}\right)^2 \approx 1\end{aligned} \tag{3.31}$$

$$\begin{aligned}&(F_I \circ F_{DF})^2\left(\frac{\partial(F_{sign} \circ F_{FRep})}{\partial p_1} + ... + \frac{\partial(F_{sign} \circ F_{FRep})}{\partial p_n}\right)^2 + \\ &+ (F_{sign} \circ F_{FRep})^2\left(\frac{\partial(F_I \circ F_{DF})}{\partial p_1} + ... + \frac{\partial(F_I \circ F_{DF})}{\partial p_n}\right)^2 + \\ &+ 2(F_{sign} \circ F_{FRep}) \cdot (F_I \circ F_{DF})\left(\frac{\partial(F_{sign} \circ F_{FRep})}{\partial p_1}\frac{\partial(F_I \circ F_{DF})}{\partial p_1} + ... + \right. \\ &\left. + \frac{\partial(F_{sign} \circ F_{FRep})}{\partial p_n}\frac{\partial(F_I \circ F_{DF})}{\partial p_n}\right) \approx 1\end{aligned} \tag{3.32}$$

We can further simplify the obtained equation:

$$\begin{aligned}&(F_I \circ F_{DF})^2||\nabla(F_{sign} \circ F_{FRep})||^2 + (F_{sign} \circ F_{FRep})^2||\nabla(F_I \circ F_{DF})||^2 + \\ &+ 2(F_{sign} \circ F_{FRep}) \cdot (F_I \circ F_{DF})\nabla(F_{sign} \circ F_{FRep})\nabla(F_I \circ F_{DF}) \approx 1\end{aligned} \tag{3.33}$$

Now let us formally show that smoothed unsigned distance function $||\nabla(F_I \circ F_{DF})||^2 \approx 1$. To do so, first, let us substitute this composition in the eikonal equation and rewrite it similarly to equation (3.30):

$$\left(\frac{\partial(F_I \circ F_{DF})(p_1)}{\partial p_1}\right)^2 + ... + \left(\frac{\partial(F_I \circ F_{DF})(p_n)}{\partial p_n}\right)^2 \approx 1 \tag{3.34}$$

After applying the chain rule, we will obtain:

$$\left(\frac{\partial(F_I(F_{DF}(p_1))}{\partial F_{DF}(p_1)}\frac{\partial F_{DF}(p_1)}{\partial p_1}\right)^2 + ... + \left(\frac{\partial(F_I(F_{DF}(p_n))}{\partial F_{DF}(p_n)}\frac{\partial F_{DF}(p_n)}{\partial p_n}\right)^2 \approx 1 \tag{3.35}$$

It can be simplified further as follows:

$$||\nabla F_{DF}(\boldsymbol{p})||^2\left(\left(\frac{\partial(F_I(F_{DF}(p_1))}{\partial F_{DF}(p_1)}\right)^2 + ... + \left(\frac{\partial(F_I(F_{DF}(p_n))}{\partial F_{DF}(p_n)}\right)^2\right) \approx 1 \tag{3.36}$$

The approximate equality will hold true if the second term will be approximately equal to one. Taking this into account, we can rewrite equation (3.33) as follows:

$$\begin{aligned}&(F_I \circ F_{DF})((F_I \circ F_{DF})||\nabla(F_{sign} \circ F_{FRep})||^2 + 2(F_{sign} \circ F_{FRep}) \times \\ &\times \nabla(F_{sign} \circ F_{FRep})\nabla(F_I \circ F_{DF})) \approx 1 - (F_{sign} \circ F_{FRep})^2\end{aligned} \tag{3.37}$$

$$(F_{sign} \circ F_{FRep}) = \begin{cases} -1, & F_{FRep} < 0 \\ 0 & F_{FRep} = 0 \\ 1, & F_{FRep} > 0 \end{cases} \tag{3.38}$$

If $(F_{sign} \circ F_{FRep}) = \pm 1$, then we can rewrite equation (3.37) as:

$$(F_I \circ F_{DF})((F_I \circ F_{DF})||\nabla(F_{sign} \circ F_{FRep})||^2 \pm 2\nabla(F_{sign} \circ F_{FRep})\nabla(F_I \circ F_{DF})) = 0 \tag{3.39}$$

The equation (3.37) will hold if one of the following conditions holds true:

$$(F_I \circ F_{DF}) = 0 \tag{3.40}$$

or

$$(F_I \circ F_{DF})||\nabla(F_{sign} \circ F_{FRep})||^2 = \mp 2\nabla(F_{sign} \circ F_{FRep})\nabla(F_I \circ F_{DF}) \tag{3.41}$$

This equality $(F_I \circ F_{DF}) = 0$ holds if the point $\boldsymbol{p}$ belongs to the boundary of the object. Otherwise, the equation (3.37) will hold true only if the equation (3.41) will be satisfied. Note that, as we have stated above, for relatively huge values of the FRep function $F_{FRep}(\boldsymbol{p})$, the composition $(F_{sign} \circ F_{FRep})(\boldsymbol{p})$ will be equal to ± 1. Thus, the gradient of the composition $\nabla(F_{sign} \circ F_{FRep})$ in equation (3.40) will be equal to zero and this equation holds. That means that we can imply an approximate equality in the eikonal equation.

If $(F_{sign} \circ F_{FRep}) = 0$, then we can rewrite equation (3.37) as:

$$(F_I \circ F_{DF})^2||\nabla(F_{sign} \circ F_{FRep})||^2 \approx 1 \tag{3.42}$$

In this case $(F_I \circ F_{DF})^2$ should be approximately close to zero, i.e. should compensate the influence of this term $||\nabla(F_{sign} \circ F_{FRep})||^2$.

Taking these assumptions into account we obtain:

$$||\nabla F_{HFRep}(\boldsymbol{p})||^2 = (F_{sign} \circ F_{FRep})^2||\nabla(F_I \circ F_{DF})||^2 \approx 1 \tag{3.43}$$

□

3.6 HFRep: Object Generation

Let us outline in a step-by-step manner the algorithmic solution for the generation of HFRep functions. The basic algorithm covers all paired combinations of FRep with DF representations, namely SDF, ADF and IDF, and describes how to generate geometric shape and specify attributes for it. Some steps of the basic algorithm will depend on the particular type of DF paired with FRep. Let us start from the basic algorithm for generating a geometric shape of the object $O_{H_{V,HFRep}}$. Fig. 3.8 demonstrates the generated function field for each step of the basic algorithm.

3.6.1 Algorithm for HFRep Geometry Generation

1. According to the definition 3.4.1, we start the construction of an HFRep object O_{HFRep} from defining the FRep function $F_{FRep}(\boldsymbol{p})$ for its geometric shape G. The FRep function $F_{FRep}(\boldsymbol{p})$ can be defined analytically, with

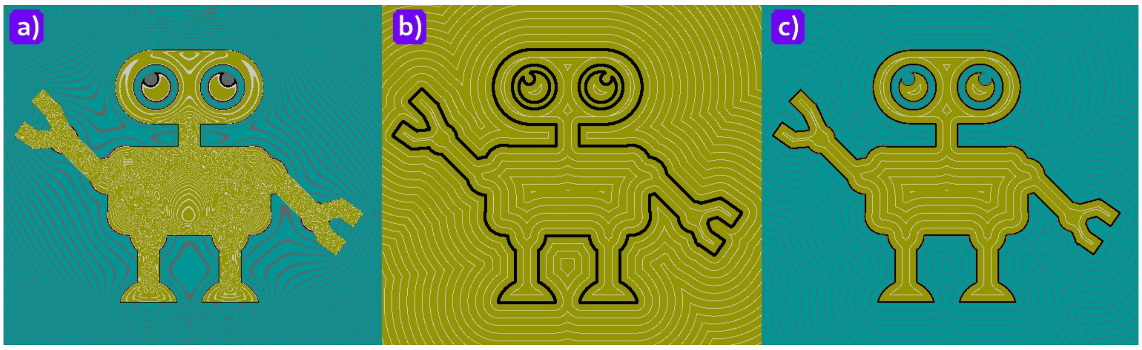


Figure 3.8: *The illustration of the basic algorithm: a) step 1: the computed field of the 'robot' FRep object; b) steps 2 - 3: the computed unsigned distance field that can be obtained using, e.g., the distance transform or a numerical solution of the eikonal equation. The obtained field is smoothed using some spline interpolation; c) step 4: the generated HFRep field.*

function evaluating algorithm or using a point cloud for which it is possible to obtain a real-valued at least C^0 continuous $F_{FRep}(\boldsymbol{p})$. It could also be a complex FRep object that is obtained in the form of a constructive tree.

At this step, we can also enforce HFRep function $F_{HFRep}(\boldsymbol{p})$ to be at least C^1 continuous as its continuity depends on the continuity of $F_{FRep}(\boldsymbol{p})$. We have to examine the obtained $F_{FRep}(\boldsymbol{p})$ for continuity and differentiability.

While processing HFRep objects, we should also consider several types of set-theoretic operations with different continuity properties. This operations belong to the class of R-functions introduced by Rvachev (Rvachev 1973). The following systems can be used: R_0, R_1 and $\overset{0}{R}$. R_0 and R_1 systems are both C_0 continuous and $\overset{0}{R}$ is C^{n-1} continuous. We will discuss them in more details in section 3.8.

Fig. 3.8 (a) shows the FRep field obtained for the 'robot' object, that was generated using 39 set-theoretic operations defined by equations (3.58) applied to circles and rectangles.

2. The values of the function $F_{FRep}(\boldsymbol{p})$ are used as an input for computing distance functions $F_{DF}(\boldsymbol{p}, \partial G)$ that should satisfy one of the definitions 3.2.2, 3.2.4 or 3.2.5. At this step we obtain an unsigned distance function that is defined as:

$$F_{DF}(\boldsymbol{p}) = d(\boldsymbol{p}, \partial G), \quad \forall \boldsymbol{p} \in X \tag{3.44}$$

Fig. 3.8 (b) shows the unsigned distance field that was obtained on the basis of a typical SDF generation algorithm SSEDT described in (Leymarie and Levine 1992).

If the distances are computed using ADF, first, we need to subdivide the space using a hierarchical data-structure, e.g. quadtree, Fig. 3.9 (b) and during its construction we also need to compute basis functions, basis vertices and extraction operators for the hierarchical splines. Then we need to compute the distances at the corner vertices of each cell. Finally, we restore distances in interior of each cell using at least C^1 continuous hierarchical spline-based

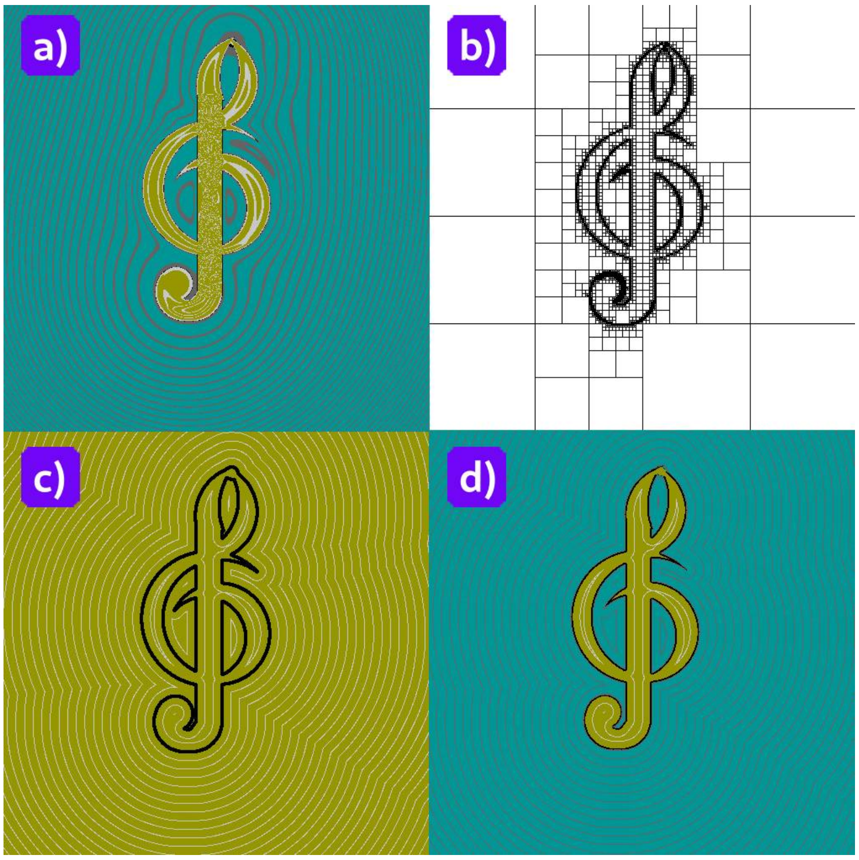


Figure 3.9: *The illustration of HFRep based on FRep and ADF with applied PHT-spline interpolation to restore the distance field at each cell. ADFs are generated using a numerical solution of the eikonal equation on the quadtree. a) the FRep field; b) a hierarchical quadtree subdivision (the maximum tree depth equals to* 10 *with* 4201 *leaves); c) UDF computed on the quadtree with the applied PHT-spline interpolation for restoring distances at each quadtree cell; d) the HFRep field that was obtained using the generated ADF.*

interpolation to obtain a smooth and continuous distance field, e.g. shown in Fig. 3.9 (c).

Specifically for IDFs, the function $F_{DF}(\boldsymbol{p})$ is defined according to equation (3.14). Distances are computed on the boundary of the object O_{FRep} and then interpolated in its interior. In Fig. 3.10 (a) we can see the field of the FRep-defined 'star' object that was used for generating HFRep IDF-based field that is shown in Fig. 3.10 (b).

In Fig. 3.11 (a) we show a possible extrapolation scheme that can be used to obtain distances in exterior of the object and make an IDF-based field signed at the last step of this algorithm. To do this, we need to use the boundary distances (see Fig. 3.11 (a), dark blue circles) and an appropriate at least C^1 continuous extrapolation operation, that will be used for obtaining distances outside the object (see Fig. 3.11 (a), red circles).

3. The distance field obtained at the previous step is unsigned and discrete as it was computed on the finite point subset $X \subset \mathbb{R}^n$. To enforce the continuity and smoothness of the computed field, we need to apply some at least C^1 continuous interpolation function $F_I(\cdot)$ to the generated unsigned field, e.g.

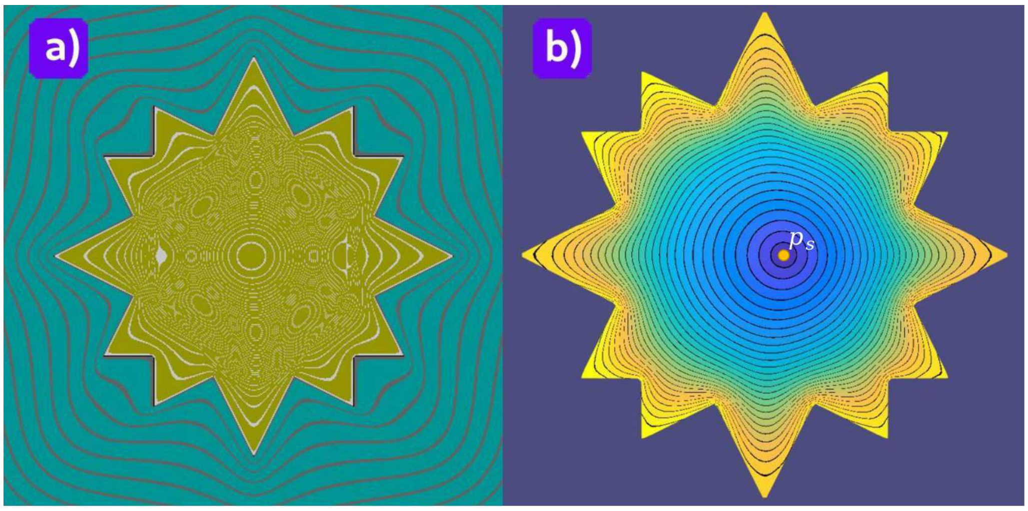


Figure 3.10: *(a) 'Star' object and its FRep field; (b) the HFRep 'star' object generated on the basis of the FRep object. The boundary of the FRep object (a) was extracted and then used for computing boundary distances. The obtained distances were interpolated in interior of the HFRep 'star' object using barycentric interpolation and mean-value coordinates. The isolines and colour show how the field changes from the source point (white circle) towards the object boundary.*

spline-based:

$$F_{smDF}(\boldsymbol{p}) = (F_I \circ F_{DF})(\boldsymbol{p}) \tag{3.45}$$

We also need to apply a smoothing operation to an IDF field if at the previous step an extrapolation operation was applied. Otherwise, IDFs are smooth as smoothness is their inherent property. An important requirement for the interpolation function $F_I(\cdot)$ is to avoid introducing extra zeros in the distance field generated using function $F_{DF}(\boldsymbol{p}, \partial G)$.

4. Finally, as the distance field obtained after previous steps is unsigned, we need to restore the field sign to distinguish between exterior $X \backslash G$, boundary ∂G and interior G_{in} of the object $O_{H_{V,HFRep}}$. We suggest to use some at least C^1 continuous step-function $F_{st}(F_{FRep}(\boldsymbol{p}))$ with the scope $[-1, 1]$, that depends on the values of the defining FRep function $F_{FRep}(\boldsymbol{p})$ and approximates its well-defined behaviour (-1 in exterior of the object, 0 on the boundary of the object and $+1$ in interior of the object). Therefore, the resulting HFRep function $F_{HFRep}(\boldsymbol{p})$ is defined according to definition 3.4.1 as follows:

$$F_{HFRep}(\boldsymbol{p}) = (F_{st} \circ F_{FRep})(\boldsymbol{p}) \cdot F_{smDF}(\boldsymbol{p}) \tag{3.46}$$

The HFRep field generated by this function can be seen in Fig. 3.8 (c). After a geometric shape of the HFRep object O_{HFRep} was generated, we can apply different operations to it, provided that they are realised by functions which are at least C^0 continuous. The HFRep object is also compatible with other distance-based objects. However, to preserve the distance properties for the object obtained after applying multiple operations, we might need to apply the steps of this algorithm again to this object.

There is a limited number of operations that preserve the distance property for the geometric shape G of the object obtained after their application. These operations are rigid (Euclidean) transformations: rotations, translations, reflections

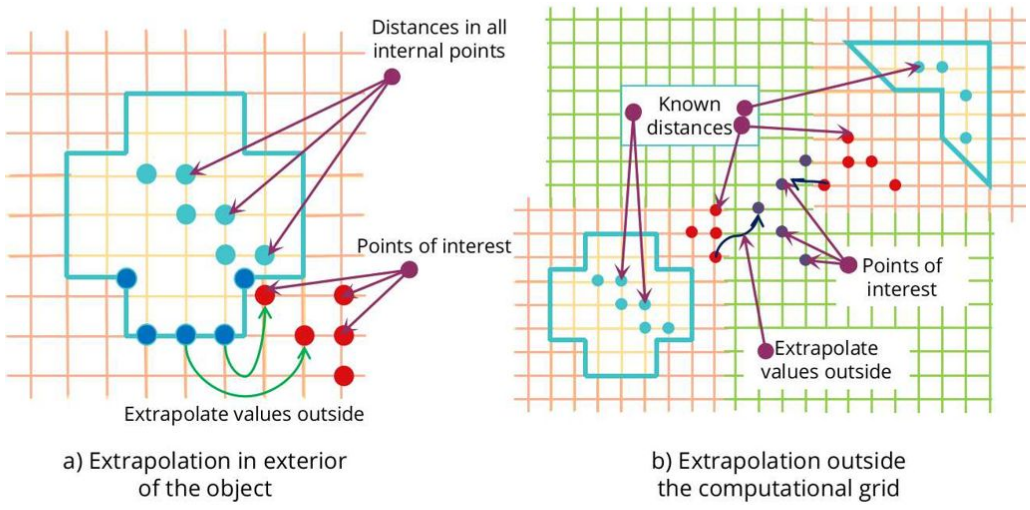

Distances in all internal points
Points of interest
Extrapolate values outside
a) Extrapolation in exterior of the object
Known distances
Points of interest
Extrapolate values outside
b) Extrapolation outside the computational grid

Figure 3.11: *Two cases when extrapolation is important to enforce the continuity of the field: a) when we have DF (light pink and light orange colours) for two objects that are distantly placed in space. In this case we need to extrapolate the distance values into the points of the green grid; b) when we computed the IDF and it is essential to obtain distances in exterior of the object.*

or their combination. Another distance preserving operations (Payne and Toga 1992) are offsetting, linear surface interpolation, surface blurring and compression, set-theoretic operations (R_α function system with $\alpha = 0$ or $\min(\cdot,\cdot)\backslash\max(\cdot,\cdot)$ operations). We will discuss them in more details in subsection 3.8.1.

In cases of other operations (Reiner, Mückl, and Dachsbacher 2011) (e.g., scaling, blending, twisting), set-theoretic operations in the form of R-functions (systems R_α with $\alpha = 1$ and $\overset{0}{R}$) (Pasko, Adzhiev, Sourin, et al. 1995)) after their application, we have to apply the basic algorithm to the obtained object to restore the distance property. These operations will be discussed in subsection 3.8.2.

To make the HFRep representation continuous on the whole domain of the Euclidean space $\mathbb{R}^n$, we suggest to apply some at least C^1 continuous extrapolation operation (e.g. (Tam and Kurbatskii 2000; Wu, Deng, and Chen 2007)) to the generated field of the object. To explain this idea in more details, let us consider the following example shown in Fig. 3.11 (b). In this figure we have two blue objects defined on their own pink grids and spaced from each other, so their defining grids are not overlapping. If we want to work with them, e.g. by applying some operation, we need somehow to define the distances in the points of interest of the green grid. One can extrapolate and average the distances between two pink grids and avoid full reinitialisation of the distances for both objects.

3.6.2 Algorithm for HFRep Attribute Definition

To set up the attributes in interior of the HFRep object $O_{H_{V,HFRep}}$, we assume that we have obtained a C^1 continuous distance function for a geometric shape. Now we can deal with the attributes that are parameterised by the distances as it was required by definition 3.4.2. Object attributes could be of different nature and there is no single algorithm to define all of them. In this work we consider such attributes as colours, microstructures, and simple 2D and volumetric textures based on noise functions parameterised by distances. Let us formulate the basic algorithm for specifying an attribute component A_i of the $O_{H_{V,HFRep}}$ on the basis of already defined geometry:

1. Depending on the nature of the attributes and how they are distributed in interior of the object $O_{H_{V,HFRep}}$, there are two possible types of object partitioning: single and multiple partitions. At this step we need to subdivide an object $O_{H_{V,HFRep}}$ according to the chosen partitioning scheme.

2. Then we specify and evaluate an attribute function $F_{A_i}(F_{HFRep}(\boldsymbol{p}), \boldsymbol{p})$ for each partition to set up the attributes at the points $\boldsymbol{p} \in G$. These functions depend on the evaluation point coordinate and are parameterised by the computed distance using $F_{HFRep}(\boldsymbol{p})$ values.

3. In case when we have a multiple partitioned object with several specified attributes, we can obtain a single attribute function for all subsets A_i by applying some interpolation, e.g. transfinite interpolation (Rvachev et al. 2001) or space-time transfinite interpolation (Sanchez, Fryazinov, Adzhiev, et al. 2015).

The more detailed discussion how to deal with attributes will be provided in section 4.6.

3.7 HFRep: Objects

The HFRep object $O_{H_{V,HFRep}}$ consists of two parts: geometric shape G and attributes A_i as it was discussed in section 3.4. The geometric shape G of the object can be obtained as a conversion of the input object to the HFRep object. The input object can be represented as follows:

- The FRep objects that are defined as algebraic solids expressed in the form of polynomials (cylinder, octahedron, sphere etc.) (Pasko and Adzhiev 2004);

- The FRep objects that are defined as skeleton-based implicit surfaces: blobby, soft objects, metaballs and convolution objects (Bloomenthal and Wyvill 1997; McCormack and Sherstyuk 1998);

- Objects that are defined as a discrete scalar fields (voxels) with trilinear or higher order interpolation (e.g. (Adzhiev, Kazakov, et al. 2000));

- The BRep objects converted to implicit surfaces or FReps (Sanchez, Fryazinov, and Pasko 2012);

- Objects that are reconstructed from scattered point clouds using radial-basis functions (e.g. (Ohtake, Belyaev, and Seidel 2006)) or similar methods that provides the defining function for the geometric shape of the object;

- Object that are defined using hypertextures, solid noise functions or extruded noise (Perlin and Hoffert 1989);

- Objects that are defined using bivariate and trivariate splines (Lai 2009).

The resulting HFRep object can be converted into other convenient representation or treated differently depending on the application. We highlight several types of the objects that can be obtained depending on the solving problem:

- It can be volume objects represented as continuous signed distance function;
- It can be converted to the BRep representation;
- It can be a heterogeneous object with geometric shape defined as a smooth continuous signed distance function;
- It can be a heterogeneous object with geometric shape and attributes (e.g. materials) defined by a smooth continuous signed distance function;
- It can be time-dependent heterogeneous objects that can be converted into time-dependent BRep objects.

The geometric shape of the HFRep object and its attribute part should be efficiently stored and processed. The HFRep object $O_{H_v,HFRep}$ can be defined using adaptive voxel representations (e.g. OpenVDB (Museth 2013) or GVDB (Hoetzlein 2016)). Tree data-structures (quadtrees, octrees (Mehlhorn 1984) etc.) or hash-tables (Zanni, Claux, and Lefebvre 2018) can be used instead as they can be efficiently stored, processed, and they are fast to rebuild. The last property is important for time-variant objects and different simulations.

3.8 HFRep: Operations

The operations that are defined for the HFRep objects can be subdivided into two groups: the ones that preserve distances and the ones that make the field of the obtained object non-distance based. For those operations that do not preserve distance properties we have to apply the basic algorithm introduced in section 3.6 to the field obtained as a result of the operation to restore the distances. We briefly outline them in this section.

3.8.1 Distance Preserving Operations

There are several operations that preserve distance property and that can be used with HFRep and other distance-based objects. They are Euclidean transformations, Affine transformations, set-theoretic operations that are defined using $\min(\cdot,\cdot)$ or $\max(\cdot,\cdot)$ functions, metamorphosis and offsetting.

Affine Transformation

An affine transformation or affine map is a geometric transformation that is defined as a function which maps an affine space onto itself while preserving both the dimension of any affine subspaces and the ratios of the lengths of parallel line segments. The distances are preserved if all affine transformations are applied in Euclidean group. Formally affine map is defined as follows:

$$f_{affine}(\boldsymbol{v}) = A\boldsymbol{v} + \boldsymbol{v}_{tr} \tag{3.47}$$

where A is an augmented matrix (affine transformation matrix: rotation, reflection), $\boldsymbol{v}$ is a vector and $\boldsymbol{v}_{tr}$ is a translation vector. If $\det(A) > 0$, the transformation is orientation-preserving, if $\det(A) < 0$, the transformation is orientation-reversing.

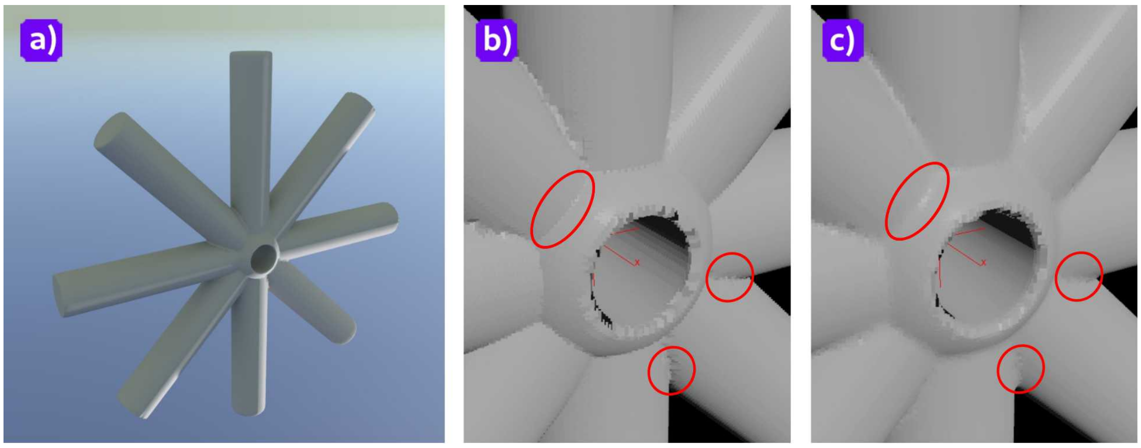


Figure 3.12: *The HFRep object (a) that was constructed using five set-theoretic operations applied to three cylinders and one sphere. b) The result of application union and subtraction set-theoretic operations in the form of R_0 R-function system (3.50) that produces C^0 continuous result. c) The result of application union and subtraction set-theoretic operations in the form of $\overset{0}{R}$ R-function system (3.59) that produces at least C^1 continuous result. The main differences in the resulting object after the application of these operations are highlighted in red areas.*

Euclidean Transformations

This type of transformations are also known as rigid transformations that are a geometric transformation of Euclidean space, which preserves the Euclidean distance between every pair of points. These operations are rotations, translations, reflections and their different combinations. A rigid transformation $f_{tr}(\boldsymbol{p}) : \mathbb{R}^n \mapsto \mathbb{R}^n$, taking into account distance preserving property, can be defined as:

$$F_{DF}(f_{tr}(\boldsymbol{p_1}), f_{tr}(\boldsymbol{p_2})) = F_{DF}(\boldsymbol{p_1}, \boldsymbol{p_2}) \tag{3.48}$$

and the Euclidean transformation is defined as:

$$f_{tr}(\boldsymbol{v}) = R(e, \theta)\boldsymbol{v} + \boldsymbol{v}_{tr} \tag{3.49}$$

where $R(e, \theta)$ is a matrix describing the rotation of the vector $\boldsymbol{v}$ around axis e and $\boldsymbol{v}_{tr}$ is a translation vector. In Fig. 3.12 (a) we have applied several Euclidean transforms to HFRep cylinders in the form of rotations. This do not violate the distance property of the field.

Set-Theoretic Operations: Min/Max

There are not a lot of variations of set-theoretic operations that are able to preserve distance property and are efficient to compute. The most commonly used set-theoretic operations for scalar fields are defined as R_α system with $\alpha = 0$ (Rvachev 1973):

$$\begin{aligned} f_\cup(f_1(\boldsymbol{p}), f_2(\boldsymbol{p})) &= 0.5(f_1 + f_2 + |f_1 - f_2|) = \max(f_1(\boldsymbol{p}), f_2(\boldsymbol{p})) \\ f_\cap(f_1(\boldsymbol{p}), f_2(\boldsymbol{p})) &= 0.5(f_1 + f_2 - |f_1 - f_2|) = \min(f_1(\boldsymbol{p}), f_2(\boldsymbol{p})) \end{aligned} \tag{3.50}$$

These operations are associative and commutative. The main disadvantage of this system is that it produces medial discontinuities in the resulting field. In Fig. 3.12 (a) we show an HFRep object that was obtained using five set-theoretic operations applied to three HFRep cylinders and one HFRep sphere. In Fig. 3.12 (b) it can be visually seen that the resulting field is not smooth, and, therefore, it is C^0 continuous.

We can formally show, that this system preserves the distance property. As we have mentioned in subsection 2.3.5, the SDF function satisfies the solution of the eikonal equation $||\nabla f(\boldsymbol{x})|| = 1$. We can rewrite the eikonal equation as:

$$\sqrt{\left(\frac{\partial f(\boldsymbol{x})}{\partial x_1}\right)^2 + ... + \left(\frac{\partial f(\boldsymbol{x})}{\partial x_n}\right)^2} = 1 \tag{3.51}$$

In distance-based modelling we are dealing with real-valued functions. Without any loss of the generalisation, we can omit squared root. Instead of the function $f(\boldsymbol{x})$, we substitute, for example, $f(f_1, f_2) \equiv f$ and obtain:

$$\left(\frac{\partial f}{\partial f_1}\frac{\partial f_1}{\partial x_1} + \frac{\partial f}{\partial f_2}\frac{\partial f_2}{\partial x_1}\right)^2 + \left(\frac{\partial f}{\partial f_1}\frac{\partial f_1}{\partial x_n} + \frac{\partial f}{\partial f_2}\frac{\partial f_2}{\partial x_n}\right)^2 = 1$$

After we raised in power of two both terms and collect its members accordingly, we obtain:

$$\left(\frac{\partial f}{\partial f_1}\right)^2\left[\left(\frac{\partial f_1}{\partial x_1}\right)^2 + ... + \left(\frac{\partial f_1}{\partial x_n}\right)^2\right] + 2\frac{\partial f}{\partial f_1}\frac{\partial f}{\partial f_2}\left[\frac{\partial f_1}{\partial x_1}\frac{\partial f_2}{\partial x_1} + ... + \frac{\partial f_1}{\partial x_n}\frac{\partial f_2}{\partial x_n}\right] + \tag{3.52}$$

$$+\left(\frac{\partial f}{\partial f_2}\right)^2\left[\left(\frac{\partial f_2}{\partial x_1}\right)^2 + ... + \left(\frac{\partial f_2}{\partial x_n}\right)^2\right] = 1$$

We can further simplify the obtained expression, taking into account equation (3.51):

$$\left(\frac{\partial f}{\partial f_1}\right)^2 ||\nabla f_1||^2 + 2\frac{\partial f}{\partial f_1}\frac{\partial f}{\partial f_2}\nabla f_1 \nabla f_2 + \left(\frac{\partial f}{\partial f_2}\right)^2 ||\nabla f_2||^2 = 1 \tag{3.53}$$

Let us assume that $||\nabla f_1||^2 = 1$ and $||\nabla f_2||^2 = 1$ that correspond to the eikonal equation. Then we obtain:

$$\left(\frac{\partial f}{\partial f_1}\right)^2 + \left(\frac{\partial f}{\partial f_2}\right)^2 + 2\frac{\partial f}{\partial f_1}\frac{\partial f}{\partial f_2}\nabla f_1 \nabla f_2 = 1 \tag{3.54}$$

To avoid dependency on the gradient term, we have to state that $\frac{\partial f}{\partial f_1}\frac{\partial f}{\partial f_2} = 0$. In this case we obtain the following system of equations:

$$\begin{cases} \left(\frac{\partial f}{\partial f_1}\right)^2 + \left(\frac{\partial f}{\partial f_2}\right)^2 = 1 \\ \frac{\partial f}{\partial f_1}\frac{\partial f}{\partial f_2} = 0 \end{cases} \tag{3.55}$$

If we substitute either $f_\cup(f_1, f_2)$ or $f_\cap(f_1, f_2)$ from equations (3.50) to the system of equations (3.55), we can see after taking into account the modulus in both of them, that they satisfy the solution of this system.

Metamorphosis

Let us assume that we have two HFRep objects $O_{HFRep,1}, O_{HFRep,2}$ with geometric shapes G_1 and G_2, and its defining functions $F_{HFRep,1}(\boldsymbol{p})$ and $F_{HFRep,2}(\boldsymbol{p})$. Then we can define the linear metamorphosis for HFRep objects as follows:

$$F_{HFRep,3}(\boldsymbol{p}, t) = F_{HFRep,1}(\boldsymbol{p})(1 - g(t)) + F_{HFRep,2}(\boldsymbol{p})g(t) \tag{3.56}$$

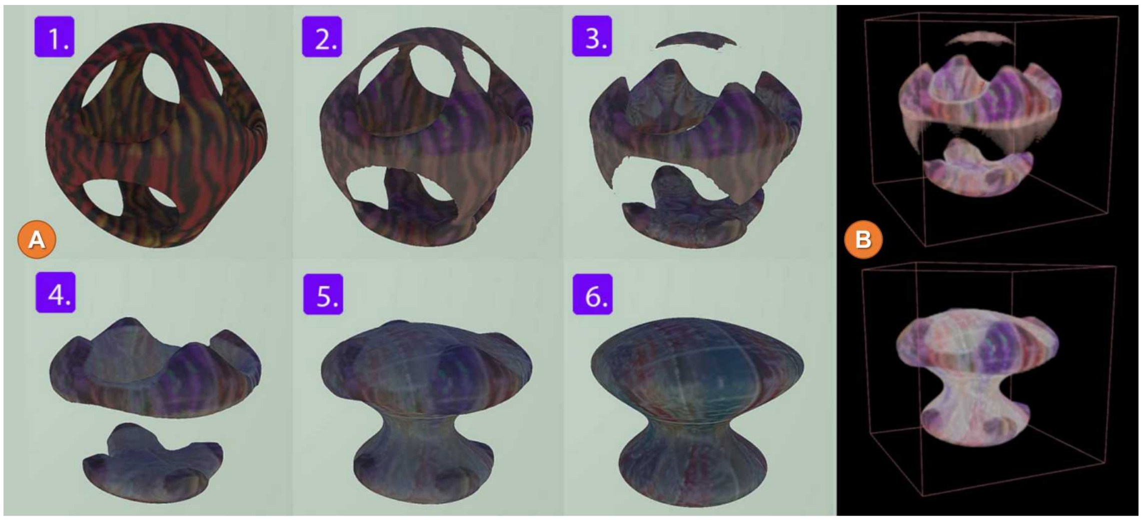


Figure 3.13: *The result of metamorphosis operation computed between an HFRep 'orthocircle' and an HFRep 'blobby' objects. A) A sequence of inbetween frames for the computed metamorphosis. B) Two frames that show the volumetric nature of the input HFRep objects. Supplementary video: figure3.13.mpg.*

where $\boldsymbol{p}$ is a multidimensional point, t parameter corresponds to time, $g(t)$ is a positively defined continuous function, that defines weighting coefficients for the linear interpolation. The domain of the function $g(t)$ corresponds to the interval $[0, 1]$: $g(t)|_{t=0} = 0$ (only the first object exists) and $g(t)|_{t=1} = 1$ (only the second object exists). It can also be formally proved that the linear metamorphosis preserves the distance in the same manner as we did it for set-theoretic operations min and max.

We can also apply this operation to the heterogeneous HFRep objects $O_{H_{V,HFRep}}$. In this case, we also need to handle attribute transformations, that can be done using, for example, one of these methods: transfinite interpolation (Rvachev et al. 2001) or space-time transfinite interpolation (Sanchez, Fryazinov, Adzhiev, et al. 2015).

In Fig. 3.13 we show a sequence of inbetween frames that were obtained as the result of linear metamorphosis operation applied to an HFRep 'orthocircle' and an HFRep 'blobby' objects with defined 3D textures. The texture transformations were handled using simple linear interpolation computed in HSV colour space. This example was implemented in *SideFX Houdini* using OpenVDB voxels data-structure. In Fig. 3.13 (B) we show two frames rendered as a fog with attributes to show that these objects are volumetric.

Offsetting

The offsetting operations in solid modelling was introduced in (Rossignac and Requicha 1986). Solid offsetting is a closed operation which means that the solid is mapped to solid. An offsetting operation expands or contracts the initial object. The iso-valued offset of the distance-based function was proved to preserve distances in (Bálint, Valasek, and Gergó 2019). The iso-offset operation is defined as follows:

$$F_{DF}(\boldsymbol{p}, G_{offset}) = F_{DF}(\boldsymbol{p}, G) - c \tag{3.57}$$

where G_{offset} is a geometric shape that was offset by the constant $c \geq 0$ and $\boldsymbol{p} \in X \backslash G_{offset}$ is a point.

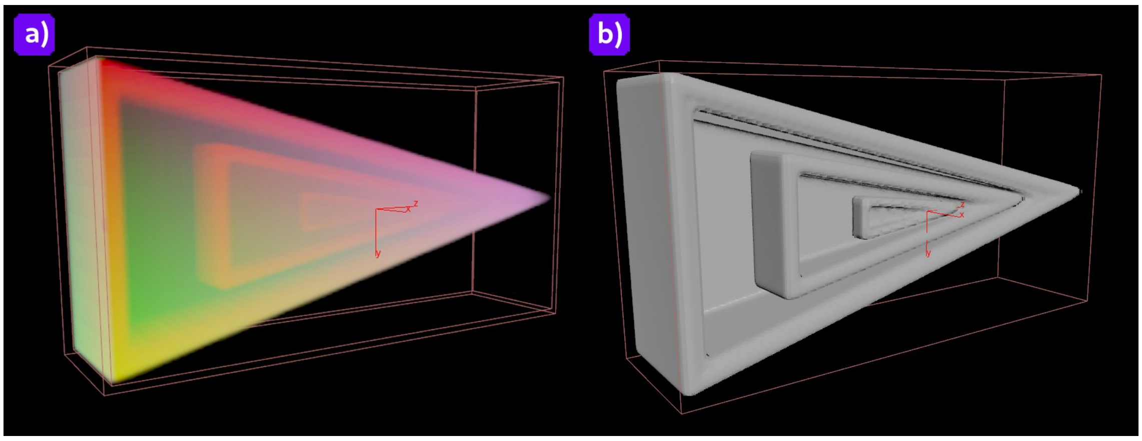


Figure 3.14: *Three halves of HFRep pyramids with different offsets applied to them to form shells with various thickness. a) the coloured OpenVDB fog that stores HFRep values; b) an HFRep object defined using OpenVDB voxels.*

In Fig. 3.14 (a) we show three coloured shelled HFRep pyramids with different surface thickness that were combined together using set-theoretic operations defined by the system of equations (3.50). To generate the shell for each pyramid object, we need to negatively offset defining function of the initial pyramid. Then, we need to subtract the result of this operation from the initial object. The resulting object can be seen in Fig. 3.14 (b).

3.8.2 Non-Distance Preserving Operations

An operations that utilise different polynomial-based operations and trigonometric functions as well as scaling operation do not preserve distances. In this case, as we discussed earlier, we have to repeat the steps of the basic algorithm for the object obtained as a result of such operations. These operations are set-theoretic operations that are based on R-function systems R_1 and $\overset{0}{R}$, scaling, twisting, blending and space-time blending.

Set-Theoretic Operations: R-function Systems R_1 and $\overset{0}{R}$

There are several R-function systems that are suitable for computing set-theoretic operations. In this subsection we cover some of them. The most practically used set-theoretic operations in the FRep modelling framework are defined as an R_α function system with $\alpha = 1$ (Pasko, Adzhiev, Sourin, et al. 1995):

$$f_\cup(f_1(\boldsymbol{p}), f_2(\boldsymbol{p})) = f_1 + f_2 + \sqrt{f_1^2 + f_2^2} \tag{3.58}$$
$$f_\cap(f_1(\boldsymbol{p}), f_2(\boldsymbol{p})) = f_1 + f_2 - \sqrt{f_1^2 + f_2^2},$$

These functions have C^1 discontinuity in points where both arguments are equal to zero. They violate the distance property. The resulting function will be C^0 continuous. If we need to obtain an at least C^1 continuous resulting function, we can apply $\overset{0}{R}$ function system that is at least C^{n-1} continuous (Rvachev 1982):

$$f_{\cup}(f_1(\boldsymbol{p}), f_2(\boldsymbol{p})) = \begin{cases} f_1 f_2 (f_1^n + f_2^n)^{-\frac{1}{n}}, & \forall f_1 > 0, f_2 > 0; \\ f_1, & \forall f_1 \leq 0, f_2 \geq 0; \\ f_2, & \forall f_1 \geq 0, f_2 \leq 0; \\ (-1)^{n+1}(f_1^n + f_2^n)^{\frac{1}{n}}, & \forall f_1 < 0; f_2 < 0; \end{cases} \tag{3.59}$$

$$f_{\cap}(f_1(\boldsymbol{p}), f_2(\boldsymbol{p})) = \begin{cases} (f_1^n + f_2^n)^{\frac{1}{n}}, & \forall f_1 > 0, f_2 > 0; \\ f_2, & \forall f_1 \leq 0, f_2 \geq 0; \\ f_1, & \forall f_1 \geq 0, f_2 \leq 0; \\ (-1)^{n+1} f_1 f_2 (f_1^n + f_2^n)^{-\frac{1}{n}}, & \forall f_1 < 0; f_2 < 0; \end{cases}$$

where $f_1(\boldsymbol{p})$ and $f_2(\boldsymbol{p})$ are any scalar field functions. This system has a better distance approximation and it possesses associative and commutative properties. In Fig. 3.12 (c) we show the result of application of these operations to three HFRep cylinders and one sphere. If we compare it with the result, shown in Fig. 3.12 (b), we can see in the highlighted areas that the field in Fig. 3.12 (c) is smooth, therefore, it is at least C^1 continuous.

Scaling

The HFRep distance can be uniformly scaled by a factor s. Formally, uniform scaling of the field is defined as:

$$F_{HFRep,s}(\boldsymbol{p}) = s \cdot F_{HFRep}(s^{-1}\boldsymbol{p}) \tag{3.60}$$

where $\boldsymbol{p}$ is a point. Unfortunately, the scaling operation violates the Euclidean distance, i.e. $||\nabla(s^{-1}\boldsymbol{p})|| = s^{-1}$.

Twisting

A twist is a mathematical operation that defines the rate of rotation of a smooth ribbon around the space curve $S_{curve} = S_{curve}(l)$, where l is the arc length of S_{curve} and $\boldsymbol{e} = \boldsymbol{e}(l)$ is a unit vector perpendicular at each point to S_{curve}. According to (Love 2013), twist operation is defined as:

$$T_w = \frac{1}{2\pi} \int \left(\frac{d\boldsymbol{e}}{dl} \times \boldsymbol{e} \right) \cdot \frac{dS_{curve}}{dl} dl \tag{3.61}$$

where $dS_{curve} \backslash dl$ is the unit tangent vector to S_{curve}. This can be rewritten in an implicit form, for example for the case, when twisting operation is applied around Z-axis, as follows:

$$\begin{aligned} x' &= x \cdot cos(\theta) + y \cdot sin(\theta); & y' &= -x \cdot sin(\theta) + y \cdot cos(\theta); \\ \theta &= (1-t)\theta_1 + t\theta_2; & t &= \frac{z - z_1}{z_2 - z_1} \end{aligned} \tag{3.62}$$

where θ_1 and θ_2 are rotation angles, z_1 and z_2 are the points of the z-interval. As this operation contains trigonometric functions it also violates the solution of the eikonal equation.

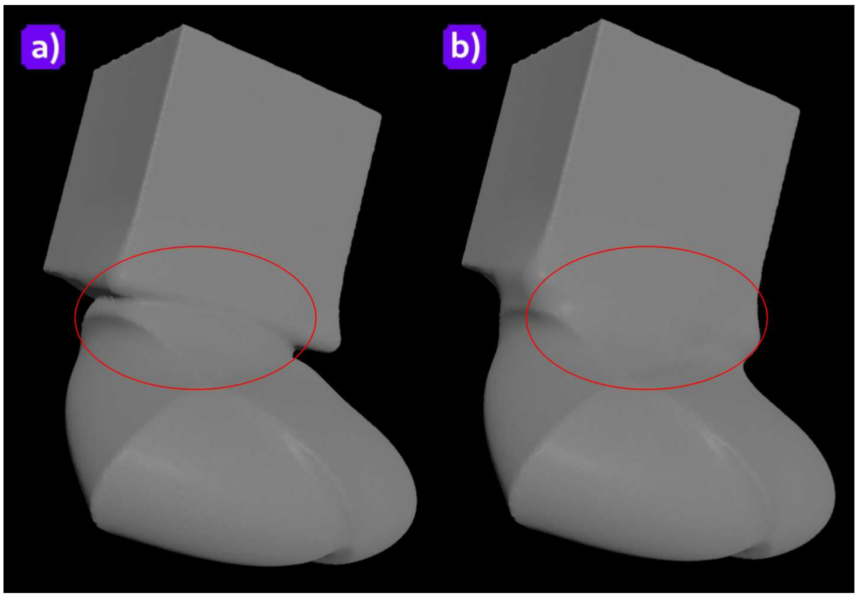


Figure 3.15: *Example of importance of C^1 continuity of the function field to avoid stresses and creases in blending.*

Blending

An analytical definition of blending that is based on R-function systems (Pasko, Pasko, and Kunii 2005) can be applied to different types of solids including distance-based. The operation of blending is an algebraic operation that produces a smoothed transition between two input objects. The blending function $F_{blend}(f_1, f_2)$ is defined as:

$$F_{blend}(f_1, f_2) = F(f_1, f_2) + disp_{blend}(f_1, f_2) \tag{3.63}$$

where $F_{blend}(f_1, f_2)$ is a blending function, $F(f_1, f_2)$ is a set-theoretic operation defined using one of the equations (3.50), (3.58) or (3.59), f_1 and f_2 are HFRep, or any other distance-based functions, or FRep, and $disp_{blend}(f_1, f_2)$ is a displacement

$$disp_{blend}(f_1, f_2) = \frac{a_0}{1 + \left(\frac{f_1}{a_1}\right)^2 + \left(\frac{f_2}{a_2}\right)^2} \tag{3.64}$$

Here $a_0, a_1, a_2 \in \mathbb{R}$ are parameters controlling the shape of the blend. The result of applying of this operation to two objects violates distance property.

In Fig. 3.15 we show the example of applying blending operation to two functionally defined objects. In Fig. 3.15 (a) the result was obtained using equation (3.63) and set-theoretic union operation defined as $max(\cdot, \cdot)$ R-function according to equation (3.50). This system of R-functions is C^1 discontinuous due to gradient discontinuities on the medial axis. In Fig. 3.15 (b) we show same blending operation, but applied with union set-theoretic operation defined as another R-function system (3.58), which becomes not differentiable only in points where both arguments are equal to zero. If we compare Fig. 3.15 (a) and (b) we could see that the result, shown in (a), is not smooth and has crease edge, shown in red circle, whilst in image (b) the transition between two shapes is smooth. Similar problems will be also observed in the case of applying the metamorphosis operation.

3.8.3 Relations

We identify several possible relations for HFRep. Their list can be extended by the user. These relations are inclusion, point membership, cell membership and intersection.

Inclusion Relation

Let we have two HFRep objects with geometric shapes G_1 and G_2 that are defined in a metric space (X, d) using functions $F_{HFRep,1}(\boldsymbol{p})$ and $F_{HFRep,2}(\boldsymbol{p})$, respectively. Then the inclusion relation $G_1 \subset G_2$ is defined as a bi-valued predicate:

$$f_{r_{in}}(\boldsymbol{p}, G_1) = \begin{cases} 0 & \text{if } F_{HFRep,1}(\boldsymbol{p}) < 0; \boldsymbol{p} \notin G_1 \\ 1 & \text{if } F_{HFRep,1}(\boldsymbol{p}) \geq 0; \boldsymbol{p} \in G_1 \end{cases} \tag{3.65}$$

where $f_{r_{in}}(\boldsymbol{p}, G_1)$ is a relation function. This relation is valid for all mentioned variations of HFRep in this work.

Point Membership Relation

Let we have an HFRep object with geometric shape G that is defined using HFRep defining function $F_{HFRep}(\boldsymbol{p})$ in a metric space (X, d). Let the boundary of the object be defined as ∂G and object's interior as $G_{in} = G \backslash \partial G$. Then we can formulate the point membership relation as:

$$f_{r_{pm}}(\boldsymbol{p}, G) = \begin{cases} 0 & \text{if } F_{HFRep}(\boldsymbol{p}) < 0; \boldsymbol{p} \notin G \\ 1 & \text{if } F_{HFRep}(\boldsymbol{p}) = 0; \boldsymbol{p} \in \partial G \\ 2 & \text{if } F_{HFRep}(\boldsymbol{p}) > 0; \boldsymbol{p} \in G_{in} \end{cases} \tag{3.66}$$

where $f_{r_{pm}}(\boldsymbol{p})$ corresponds to point membership function. The point membership written in this form is valid for HFRep that is based on SDF, ADF and IDF (if the distances were extrapolated in exterior of the object). Otherwise, the point membership for HFRep based on IDF consists only of two cases: when the point $\boldsymbol{p}$ belongs to the surface ∂G, and when the point $\boldsymbol{p}$ belongs to the interior of the object G_{in}.

When we are working with HFRep based on FRep and ADF, we can also introduce a cell point membership as the Euclidean space is subdivided into multiple nodes with cells using tree hierarchical structure. Let the cell geometry G_c be defined as a quad or cube with a specified coordinate of one of the corners $\boldsymbol{p}_{c_i}$ and the side l size. Then we can identify whether current point $\boldsymbol{p}$ belongs to the cell interior $G_{c_{in}}$, its boundary ∂G_c or its exterior:

$$f_{r_{pm}}(\boldsymbol{p}, G_c) = \begin{cases} 1 & \text{if } \boldsymbol{p} \in \partial G_c \\ 2 & \text{if } \boldsymbol{p}_{c_i} < \boldsymbol{p} < \boldsymbol{p}_{c_i} + l \Leftrightarrow \boldsymbol{p} \in G_{c_{in}} \\ 0 & \text{otherwise} \end{cases} \tag{3.67}$$

Intersection Relation

Let we have two HFRep objects with geometric shapes G_1 and G_2 that are defined using HFRep functions $F_{HFRep,1}(\boldsymbol{p})$ and $F_{HFRep,2}(\boldsymbol{p})$ in a metric space (X, d).

The intersection relation provides information about the presence of common points between two geometric shapes. Formally, it is defined as:

$$f_{r_{int}}(G_1, G_2) = \begin{cases} 0 & \text{if } F_{HFRep,1}(\boldsymbol{p}) \cap F_{HFRep,2}(\boldsymbol{p}) < 0 \Leftrightarrow G_1 \cap G_2 = \emptyset \\ 1 & \text{if } F_{HFRep,1}(\boldsymbol{p}) \cap F_{HFRep,2}(\boldsymbol{p}) \geq 0 \Leftrightarrow G_1 \cap G_2 \neq \emptyset \end{cases} \tag{3.68}$$

This property is useful while solving collision detection problems that are based on the search of the maximum $F_{HFRep,1}(\boldsymbol{p}) \cap F_{HFRep,2}(\boldsymbol{p})$. This relation works for all discussed combinations of FRep and SDF, ADF and IDF representations.

3.9 Conclusions

In this chapter we have introduced a theoretical framework for modelling volumetric heterogeneous objects on the basis of a novel unifying functionally-based hybrid representation called hybrid function representation (HFRep). In the previous chapter we have identified four conventional representations related to scalar fields of different kinds, namely FRep, SDF, ADF and IDF and identified their advantages and drawbacks. In this chapter we have proposed a formalisation of those representations and introduced formal definitions for FRep, ADF and IDF representations. This has allowed us to formulate the requirements for the unifying hybrid representation. The core of HFRep is FRep, which is coupled with one of the distance-based representations, namely SDF, ADF or IDF.

The defining functions in the core of HFRep are continuous and have a distance property everywhere in a Euclidean space. We have proven that the continuity of the HFRep function depends on the continuity of the FRep function and it is either C^0 or C^1. We have proven that the gradient of the HFRep function satisfies the eikonal equation. This means that the HFRep function preserves Euclidean distance property. We have defined the mathematical basics of the representation and developed an algorithmic procedure allowing to generate HFRep objects both in terms of their geometry and attributes. We have identified and described the objects supported by the HFRep representation. Another important component of the representation is the operations specified for the defined objects. We have identified two groups of the operations. The operations in the first group preserve Euclidean distance property while the operations in the second group violate it.

The theoretical framework presented in this chapter has been published as a preprint (Tereshin, Pasko, et al. 2020) in arXive. A full-scale paper was published in Graphical Models journal (Tereshin, Pasko, et al. 2021).

In the next chapter we will provide a detailed description of the algorithmic and technical implementation of the basic algorithm provided in this chapter.

Chapter 4

Hybrid Function Representation: Algorithmic Framework

In the previous chapter we have introduced a theoretical hybrid framework for heterogeneous HFRep object modelling that unifies advantages of FRep, SDF, ADF and IDF and compensates for their drawbacks. We have described the basic algorithm for generating three different hybrid combinations of these representations, namely, FRep and SDF, FRep and ADF, FRep and IDF. The first step of the basic algorithm has been already discussed in section 3.6. In the next sections we discuss steps 2 - 4.

For each HFRep combination we have to apply different algorithms for generating unsigned distance field at the second step of the basic algorithm. First, we describe algorithmic solutions for computing distances in the case, when HFRep is based on FRep and SDF (section 4.1) using several well-established methods. Then we introduce a detailed description of the hierarchical fast iterative method, proposed in this work, for computing at least C^1 continuous unsigned ADFs (section 4.2). Finally, we describe an algorithmic solution for generating HFRep based on FRep and IDF (section 4.3) that is based on the method described in the work (Rustamov, Lipman, and Funkhouser 2009).

As the unsigned distances obtained at the second step of the basic algorithm are computed in the finite number of points, distances are discrete and non-smooth. To make them smooth and continuous in section 4.4 we briefly discuss several techniques that can be used in this context.

To make the generated field signed according to the step four of the basic algorithm, we give several smooth step-functions in section 4.5.

In the last section of this chapter we give several representative examples that demonstrate how the proposed theoretical and practical framework works with heterogeneous objects. We briefly discuss our *C++* implementation of the described algorithms in this chapter and give a detailed description of our implementation of the HFRep framework based on FRep and SDF in *SideFX Houdini* (Side Effects Software 2020).

4.1 Algorithms for Generating SDF

In this section we consider the step two of the basic algorithm (section 3.6) with respect to generating SDF. The generation procedure for the unsigned distance

Part of this chapter was published in (Tereshin, Adzhiev, Fryazinov, and Pasko 2019; Tereshin, Pasko, et al. 2020, 2021).

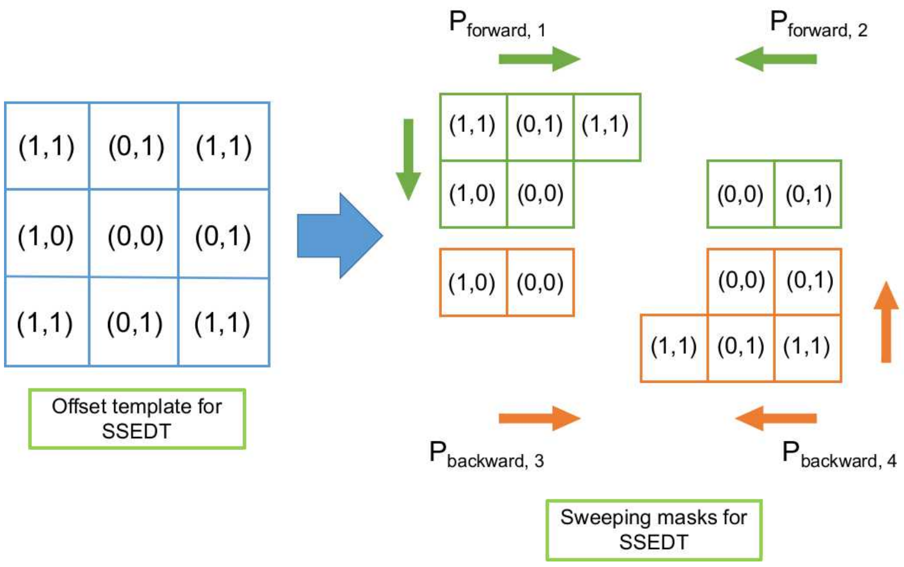


Figure 4.1: *Offset template and sweeping masks for the 8-point Signed Sequential Euclidean Distance Transform.*

field (UDF) is implemented using three different methods: the signed sequential Euclidean distance transform (SSEDT) (Leymarie and Levine 1992) for 2D case, the vector-city vector distance transform (VCVDT) (Satherley and Jones 2001) for 3D case and the fast iterative method (FIM) (Jeong and Whitaker 2008) for both 2D and 3D cases. Let us briefly outline them.

4.1.1 Distance Transform Methods

The computation of any distance transform starts from the initialisation of the computational grid. SSEDT in 2D and VCVDT in 3D algorithms are vector-based, i.e. distances are defined as magnitudes of the vector values. SSEDT provides an SDF field as an output and VCVDT provides an UDF field as an output if it is implemented according to (Satherley and Jones 2001). In our implementation we have made VCVDT algorithm to output the SDF field instead of the UDF field by following the logic of the implementation of the SSEDT algorithm.

First, we need to initialise two computational vector grids G_1 and G_2 of equal size according to Algorithm 1. We will store there vector components of the vector distance fields $\overrightarrow{\mathrm{F}}_{UDF_1}$ and $\overrightarrow{\mathrm{F}}_{UDF_2}$ with their $||L||_2$ norm computed as:

$$||\overrightarrow{\mathrm{F}}_{UDF_i}(\overline{x},\overline{y},\overline{z})|| = \sqrt{F^2_{UDF_{\overline{x},i}} + F^2_{UDF_{\overline{y},i}} + F^2_{UDF_{\overline{z},i}}} \quad i = 1, 2,$$

where $\overline{x}, \overline{y}$ and $\overline{z}$ are directions. The initialisation operation for both 2D and 3D distance transform algorithms is the same. We initialise grid points $\boldsymbol{p}_i$ of G_1 with zeroes where the FRep function is $F_{FRep}(\boldsymbol{p}_i) \geq 0$. The rest of the points are initialised with some relatively huge value. The grid points $\boldsymbol{p}_i$ of G_2 are initialised similarly, but in an inverse manner.

Then, in case of the SSEDT computation, we need to apply four passes of the offsetting distance templates (see Fig. 4.1) to both grids G_1 and G_2. First, we need to compute two forward passes $P_{forward,1}$ and $P_{forward,2}$ of the offsetting distance templates. Each pass is applied to both computational grids G_1 and G_2 in left to right, top to bottom directions. Each pass offsets the vector values stored at each point of both grids. Then we need to compute two backward passes $P_{backward,3}$ and

Algorithm 1: Signed vector distance transform

```
Input: two equal sized computational vector grids $G_1$, $G_2$; scalar grid $G_3$; a
       FRep field $F_{FRep}(\boldsymbol{p}_i)$; an UDF field $F_{UDF}(\boldsymbol{q}_i), \boldsymbol{q}_i \in G_3$
Output: $F_{UDF}(\boldsymbol{q}_i)$
/* Initialisation for SSEDT and VCVDT                                          */
1  for all points $\boldsymbol{p}_i \in G_1$ do
2      if $F_{FRep}(\boldsymbol{p}_i) \geq 0$ then
3          $F_{UDF,1}(\boldsymbol{p}_i) \leftarrow 0$;
4      else
5          $F_{UDF,1}(\boldsymbol{p}_i) \leftarrow \infty$;
6  for all points $\boldsymbol{p}_i \in G_2$ do
7      if $F_{FRep}(\boldsymbol{x}) \geq 0$ then
8          $F_{UDF,2}(\boldsymbol{p}_i) \leftarrow 0$;
9      else
10         $F_{UDF,2}(\boldsymbol{p}_i) \leftarrow \infty$;
/* Start iterative computation of the DF                                        */
11 if SSEDT then
12     Compute four passes of the distance matrix template shown in Fig. 4.1
        for both grids $G_1$ and $G_2$;
13 if VCVDT then
14     Compute eight passes of the distance matrix template shown in Fig. 4.2
        for both grids $G_1$ and $G_2$;
/* At each pass compare $||\overrightarrow{\mathrm{F}}_{UDF_k}(\boldsymbol{p}_i)||, k = 1, 2$ of the traversing grid
   ($G_1$ or $G_2$) with its offset neighbours $||\overrightarrow{\mathrm{F}}_{UDF_k}(\boldsymbol{p}_i + offset)||$      */
15 if $||\overrightarrow{\mathrm{F}}_{UDF_k}(\boldsymbol{p}_i)|| > ||\overrightarrow{\mathrm{F}}_{UDF_k}(\boldsymbol{p}_i + offset)||$ then
16     $\overrightarrow{\mathrm{F}}_{UDF_k}(\boldsymbol{p}_i) = \overrightarrow{\mathrm{F}}_{UDF_k}(\boldsymbol{p}_i + offset)$; $k = 1, 2$;
/* Compute signed distance field and store it in one dimensional
   grid $G_3$ of the same size as grids $G_1$ and $G_2$                           */
17 for all points $\boldsymbol{q}_i \in G_3$ do
       /* To make $F_{UDF}(\boldsymbol{q}_i)$ signed, remove $abs(\cdot)$                  */
18     $F_{UDF}(\boldsymbol{q}_i) = abs(||\overrightarrow{\mathrm{F}}_{UDF_1}(\boldsymbol{p}_i)|| - ||\overrightarrow{\mathrm{F}}_{UDF_2}(\boldsymbol{p}_i)||)$;
```

$P_{backward,4}$ of the offsetting distance templates in the reverse directions in the same manner.

In case of the VCVDT computation, we need to apply eight passes of the offsetting distance templates (see Fig. 4.2) to both grids G_1 and G_2. First, we need to apply four forward passes $P_{forward,1}$, $P_{forward,2}$, $P_{forward,3}$ and $P_{forward,4}$ of the offsetting distance templates. They should be applied to both computational grids G_1 and G_2 according to the arrows shown in Fig. 4.2. After that, we need to apply four backward passes $P_{backward,1}$, $P_{backward,2}$, $P_{backward,3}$ and $P_{backward,4}$ of the offsetting distance templates to both computational grids G_1 and G_2. The direction of their application is shown in Fig. 4.2.

For both algorithms SSEDT and VCVDT, at each pass we compare the distance $||\overrightarrow{\mathrm{F}}_{UDF_i}(\boldsymbol{p})||$ stored in the current point of the grid G_1 or G_2 with its offset neighbours $||\overrightarrow{\mathrm{F}}_{UDF_i}(\boldsymbol{p} + offset)||$. We store the minimum vector distance

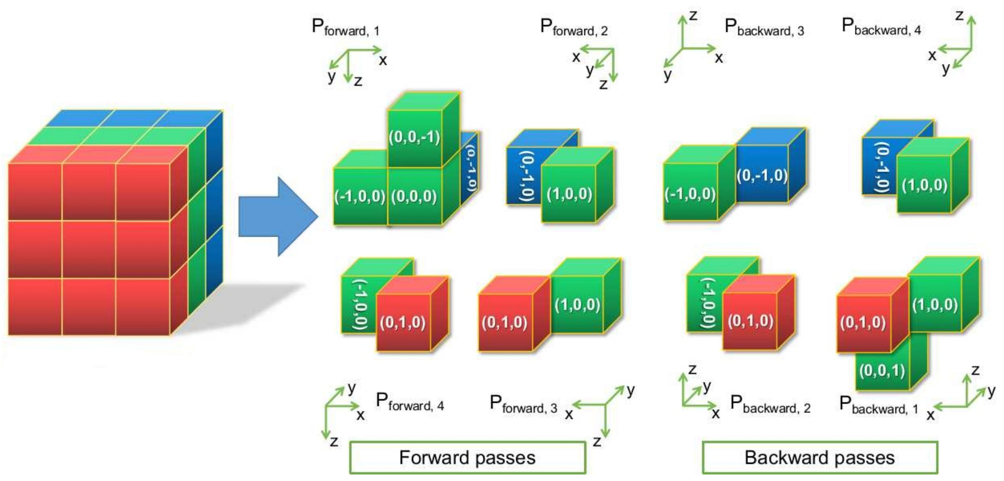


Figure 4.2: *Four offsetting distance templates for the vector-city vector distance transform that form forward and backward passes. These templates are used to offset the computing distance field values.*

$\min(||\overrightarrow{\mathrm{F}}_{UDF_i}(\boldsymbol{p})||, ||\overrightarrow{\mathrm{F}}_{UDF_i}(\boldsymbol{p} + offset)||)$ in the current point $\boldsymbol{p}$ of the grid G_1 or G_2.

Finally, to obtain SDF we compute the difference between $||L||_2$ norms of two vector fields $\overrightarrow{\mathrm{F}}_{UDF_1}(\boldsymbol{p})$ and $\overrightarrow{\mathrm{F}}_{UDF_2}(\boldsymbol{p})$ stored in grids G_1 and G_2. In our case, we compute the absolute value of the difference between two distance norms to obtain UDF, that will be further used for the generation of HFRep.

4.1.2 Fast Iterative Method

The main idea of the FIM method (Jeong and Whitaker 2008) is to solve the eikonal equation selectively on grid nodes using the data structure, which is computationally efficient, namely *active list* L (linked list). Let us recall the definition of the eikonal equation that is used for solving wave propagation problems:

$$|\nabla f(\boldsymbol{p})| \cdot f_{sp}(\boldsymbol{p}) = 1$$

where $f_{sp}(\boldsymbol{p})$ is a speed function, $f(\boldsymbol{p})$ is a distance function if $f_{sp}(\boldsymbol{p}) = 1$.

As the relation between grid nodes was made to be loose, all the nodes in active list can be updated simultaneously utilising the parallel architecture. During each iteration, new elements are added to the active list and the band expands to include all nodes that can be affected by the current update. A node can be removed from the list if it is converged to some value with respect to the threshold condition, and can be reinserted in the active list if any upwind neighbour‘s value is updated.

According to the Algorithm 2 (Hong and Jeong 2017), first, we need to do the initialisation step and define the boundary conditions. The boundary conditions consists of pre-initialisation of the solution of the eikonal equation $U(\boldsymbol{x})$ at each grid point $\boldsymbol{x} = (i, j, k)$ and adding ’source’ nodes to the active list L.

As we are dealing with FRep objects defined by the functions, we need to extract the boundary of the FRep object. The initial solution $U(\boldsymbol{x})$ is computed with respect to the grid nodes $\boldsymbol{x}$ that are associated with the boundary points of the FRep object. These nodes are initialised to zero. This means that they are effectively defined as ’source’ nodes. As all computations are done on a discrete computational grid, the defining FRep function of the object usually does not have exact zeros

Algorithm 2: Fast iterative method

```
Input: grid $G$, solution $U(\boldsymbol{x})$, active list L
Output: $U(\boldsymbol{x})$
/* Initialisation */
1  for all nodes $\boldsymbol{x} \in G$ do
2      if $0 \leq F_{FRep}(\boldsymbol{x}) \leq \epsilon$ then
3          $U(\boldsymbol{x}) \leftarrow 0$;
4          add $\boldsymbol{x}$ to L;
5      else
6          $U(\boldsymbol{x}) \leftarrow \infty$;
/* Updating nodes in active list L */
7  while active list L is not empty do
8      for all nodes $\boldsymbol{x} \in L$ do
9          $U_p \leftarrow U(\boldsymbol{x})$;
10         $U_q \leftarrow$ solution of $f_{Godunov}(\boldsymbol{x}) = 0$;
           /* If the solution is not converged */
11         if $U_p > U_q$ then
12             $U(\boldsymbol{x}) \leftarrow U_q$;
13         else
               /* Check adjacent nodes $\boldsymbol{x}_{nb}$ if they need further updates */
14             for all neighbours $\boldsymbol{x}_{nb}$ of $\boldsymbol{x}$ do
15                 if $U(\boldsymbol{x}_{nb}) > U(\boldsymbol{x})$ and $\boldsymbol{x}_{nb} \notin L$ then
16                     $U_p \leftarrow U(\boldsymbol{x}_{nb})$;
17                     $U_q \leftarrow$ solution of $f_{Godunov}(\boldsymbol{x}_{nb}) = 0$;
18                     if $U_p > U_q$ then
19                         $U(\boldsymbol{x}_{nb}) \leftarrow U_q$;
20                         add $\boldsymbol{x}_{nb}$ to $L$;
```

at the boundary points. We manually define some small interval of values in the neighbourhood of zero $0 \leq F_{FRep}(\boldsymbol{x}_b) \leq \epsilon$, where $\epsilon > 0$ is a relatively small value. Then all grid nodes that are defined as 'source' nodes should be added to the active list.

After this, the process of updating the grid nodes begins. During this stage for every node $\boldsymbol{x}$ in the active list L, the new $U(\boldsymbol{x})$ is computed using the first order upwind Godunov discretisation scheme that can be implemented using Algorithm 3 in 2D case (Gómez et al. 2019) or Algorithm 4 in 3D case (Jeong and Whitaker 2008). To check if the computed solution $U(\boldsymbol{x})$ is converged, its values are compared with the previous solution computed at the node $\boldsymbol{x}$. If it is converged, node $\boldsymbol{x}$ is removed from the active list L, and any neighbour nodes are added to active list L if they are not in it.

4.2 Algorithm for Generating ADF

In this section we consider the step two of the basic algorithm (section 3.6) with respect to generating ADF. Here we discuss how distance field can be numerically

Algorithm 3: Godunov discretisation scheme in 2D (Gómez et al. 2019)

```
Input: current solution $U(x_{i,j})$, step $h$ along the grid $G$, scalar value $f_{sp}$ of
       the speed function. Here we assume that $f_{sp} = 1$.
Output: $U_{new}(x_{i,j})$
/* Compare stored $U$ in the neighbour nodes of the current node $x_{i,j}$
   along $X$ and $Y$ directions and find the minimum one.                  */
1 $U_X = \min(U(x_{i-1,j}), U(x_{i+1,j}))$;
2 $U_Y = \min(U(x_{i,j-1}), U(x_{i,j+1}))$;
3 if $|U_X - U_Y| \leq \frac{h}{f_{sp}}$ then
4     $U_{new}(x_{i,j}) = \frac{1}{2}(U_X + U_Y + \sqrt{\frac{2h^2}{f_{sp}^2} - (U_X - U_Y)^2})$;
5 else
      /* Linear update                                                    */
6     $U_{new}(x_{i,j}) = \min(U_X, U_Y) + \frac{h}{f_{sp}}$;
```

Algorithm 4: Godunov discretisation scheme in 3D (Jeong and Whitaker 2008)

```
Input: current solution $U(x_{i,j,k})$, step $h$ along the grid $G$, scalar value $f_{sp}$
       of the speed function. Here we assume that $f_{sp} = 1$.
Output: $U_{new}(x_{i,j,k})$
/* Compare stored $U$ in the neighbour nodes of the current node $x_{i,j,k}$
   along $X, Y$ and $Z$ directions and find the minimum one.              */
1 $U_X = \min(U(x_{i-1,j,k}), U(x_{i+1,j,k}))$;
2 $U_Y = \min(U(x_{i,j-1,k}), U(x_{i,j+1,k}))$;
3 $U_Z = \min(U(x_{i,j,k-1}), U(x_{i,j,k+1}))$;
4 $U_{new}(x_{i,j,k}) = U_Z + \frac{h}{f_{sp}}$;
5 if $U_{new}(x_{i,j,k}) \leq U_Y$ then
6     return $U_{new}(x_{i,j,k})$;
7 $U_{new}(x_{i,j,k}) = \frac{1}{2}(U_Y + U_Z + \sqrt{\frac{2h^2}{f_{sp}^2} - (U_Y - U_Z)^2})$;
8 if $U_{new}(x_{i,j,k}) \leq U_X$ then
9     return $U_{new}(\boldsymbol{x}_{i,j,k})$;
10 $U_{new}(x_{i,j,k}) =$
     $\frac{1}{6}(2(U_X + U_Y + U_Z) + \sqrt{4(U_X + U_Y + U_Z)^2 - 12(U_X^2 + U_Y^2 + U_Z^2 - \frac{h^2}{f_{sp}^2}}))$;
```

computed on adaptive Cartesian grids. We introduce an adaptation of the eikonal solver, namely fast iterative method (FIM) (Jeong and Whitaker 2008) (for more details see subsection 4.1.2), that we extend for computing distances on irregular hierarchical grids. We call this extended method as a hierarchical fast iterative method (HFIM). We implement the proposed method only for 2D case.

The algorithmic implementation of HFIM consists of several steps. The first step is the hierarchical subdivision of the input object defined by the FRep function using quadtree (see subsection 4.2.3). This step also covers the construction of the basis functions for the interpolating PHT-spline surface that is at least C^1 continuous. A PHT-spline surface is specified as a tensor product of PHT-splines defined in a parametric space (see subsection 4.2.6). The second step is the computation of the HFIM method on quadtree (see subsection 4.2.2). The last step is the restoration of the computed unsigned distance field using the interpolating PHT-spline surface generated on the previous step.

4.2.1 Hierarchical FIM (HFIM): Requirements

Distance-based scalar fields are widely used in geometric modelling and related tasks for defining objects in a predictable way. We would like to introduce an exact method for efficient computation of smooth distance fields on the adaptive grid that provides at least C^1 continuous distance field. It should satisfy the following requirements:

1. The algorithm should provide an independent node update, i.e. independent update of the solution at each point of the computational grid (inherited from the original FIM method);
2. The algorithm should rely on non-heterogeneous data-structure that is efficient for sorting, inserting and removing new data (inherited from the original FIM method);
3. The algorithm should be applicable to non-graded adapted grids, i.e. the difference of levels between adjacent cells is unconstrained;
4. The algorithm should produce at least C^1 continuous distance field;

The first requirement is essential for possible paralleling of computations on the streaming architectures and cache coherency. Controlling the updating process of the algorithm is essential for updating data stored in cache (e.g. local on the grid). The second requirement is essential for efficient fit of the computed data in SIMD memory or streaming architecture memory while keeping the updates of the stored solution independent. These two requirements are inherited from the FIM method. The third requirement is essential for performing independent computations on non-graded hierarchical grids. The algorithm should be able to provide the solution of the eikonal equation on the grid with different steps along the axis. The last requirement is important for geometric modelling, especially in the case of heterogeneous object modelling, where the attributes might depend on distance values. According to the work (Biswas, Shapiro, and Tsukanov 2004), lack of differentiability of the distance field leads to the stresses and creases in the modelling attributes when they are parameterised by the distance.

4.2.2 HFIM: Algorithm

The adapted 2D HFIM algorithm follows the logic of the FIM algorithm for regular grids (see Algorithm 2). In this work we consider a FRep object defined (Pasko, Adzhiev, Sourin, et al. 1995) by the function $F_{FRep}(\boldsymbol{p})$ as an input for the HFIM Algorithm 5. The major requirement for $F_{FRep}(\boldsymbol{p})$ function is to be at least C^0 or C^1 continuous.

Let us outline the main steps of the HFIM algorithm. It consists of two parts: initialisation of the hierarchical computational grid H_G in the form of a quadtree and computation of the nodes updates. We denote corner vertices of each quadtree cell as a node $x_{i,j}$, where i is the index of the leaf in the leaf array, and j is the index of the corner vertex of the cell. The initialisation algorithm is defined as follows:

1. Let the object O_{FRep} be defined by a FRep function $F_{FRep}(\boldsymbol{p})$ computed on a regular grid. In Fig. 4.3 (a) we show the FRep field of 'i' object that serves as an input for the HFIM algorithm.

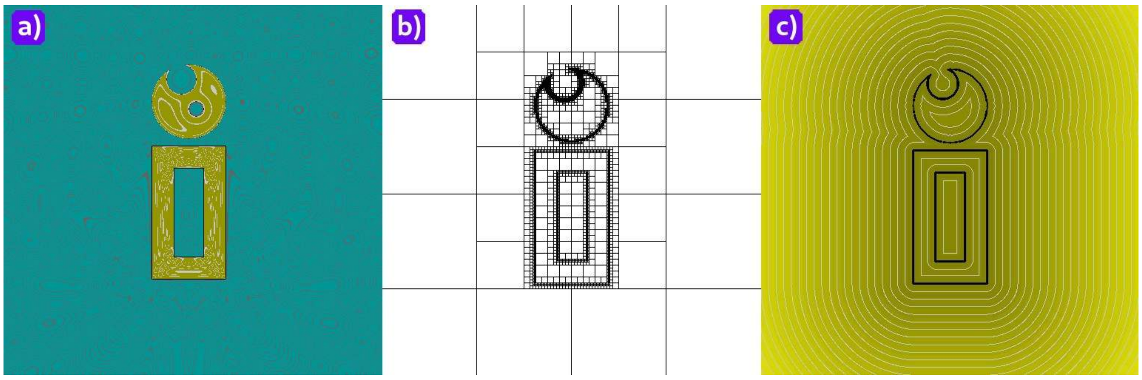


Figure 4.3: *The illustration of the main main computational steps of the HFIM method. a) The FRep field of the 'i' object computed on the regular grid. b) a quadtree obtained for the 'i' FRep object using quadtree. c) The computed unsigned distance field with applied PHT-spline interpolation to restore the distance at each quadtree cell.*

2. After we have defined the FRep object O_{FRep}, we subdivide the Euclidean space according to the defining function $F_{FRep}(\boldsymbol{p})$: a sparse amount of cells in exterior $F_{FRep}(\boldsymbol{p}) < 0$ of the object, a dense amount of cells on the boundary $F_{FRep}(\boldsymbol{p}) = 0$ of the object, and an average amount of cells in interior $F_{FRep}(\boldsymbol{p}) > 0$ of the object. This can be achieved using quadtree hierarchical data-structure.

3. At each node $x_{i,j}$ of each leaf we store the corresponding value of the FRep function $F_{FRep}(x_{i,j})$. The result of the quadtree subdivision of the Euclidean space according to the FRep function $F_{FRep}(x_{i,j})$ can be seen in Fig 4.3 (b).

4. After the hierarchical subdivision is completed, we apply the boundary conditions for solving the eikonal equation on it. We follow the FIM initialisation scheme. We traverse the quadtree and initialise corresponding nodes $x_{i,j}$ according to the following rule:

$$U(x) = \begin{cases} 0 & \text{if } 0 \le F_{FRep}(x_{i,j}) \le \epsilon \\ \text{inf} & \text{otherwise} \end{cases} \tag{4.1}$$

   where $\epsilon \in \mathbb{R}$ is a relatively small value that helps to isolate the zero level-set of the FRep object O_{FRep}, and inf is a relatively huge number.

5. We store each node that is defined as a 'source' ($U(x_{i,j}) = 0$) in the active list L as a pair of indices (i, j). In our implementation the active list is defined using *std::list* data structure.

After the initialisation of the hierarchical grid H_G, we start an iterative computation of the solution of the eikonal equation on it. The iterative process stops when the active list L is empty. For each pair of indices (i, j) stored in the active list L, we compute an updated solution according to the following steps that correspond to the Algorithm 5:

1. First, we obtain the solution $U_G(x_{i,j})$ of the the first order Godunov upwind descritisation scheme $f_{Godunov}(x_{i,j}) = 0$ for the node $x_{i,j}, (i, j) \in L$ that is written for the adaptive grid computations.

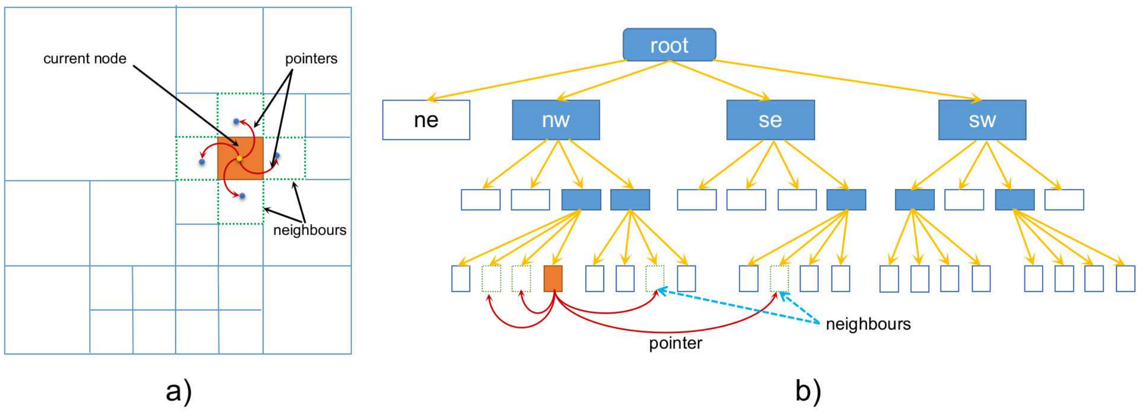


Figure 4.4: *T-mesh in the form of the quadtree. a) The subdivided space with visualisation of the relationship between current node and its neighbours. b) The tree structure of the T-mesh shown in part (a) with assigned pointers form the current node to its neighbours. Coloured rectangles show that they have children nodes and white rectangles show that they are leaves of the tree.*

2. Then we compare $U_G(x_{i,j})$ with the previously computed solution $U(x_{i,j})$ in the node $x_{i,j}$. If $U(x_{i,j}) > U_G(x_{i,j})$, we update the value in the current node $x_{i,j}$ of the tree and in the nodes that coincide with the current one.

3. If $U(x_{i,j}) < U_G(x_{i,j})$, we start traversing the quadtree to find the neighbour nodes $x^{(nb)}_{m,n}$ of the current node $x_{i,j}$ in the horizontal and vertical directions. If at this step we have found a T-junctional node in one of the directions, we need to compute a 'ghost' value as it was suggested in subsection 4.2.5. In our implementation at this step we return a pair $((m,n), U(x_{m,n}))$ of two indices and currently stored solution $U(x_{m,n})$ of the eikonal equation. The first index m points at the leaf array and the second index n points at the cell vertices array.

4. For all found neighbours $x^{(nb)}_{m,n}$ we check if the stored solution $U(x^{nb}_{m,n})$ is greater then current solution $U(x_{i,j})$. We also want to avoid unnecessary computations by checking if the found neighbour node $x^{(nb)}_{m,n}$ is already presented in the active list L or not.

5. Then we compute the solution $U_G(x^{(nb)}_{m,n})$ of the first order Godunov upwind discretisation scheme $f_{Godunov}(x^{(nb)}_{m,n}) = 0$ in the current neighbour node $x^{(nb)}_{m,n}$.

6. If the solution $U_G(x^{(nb)}_{m,n})$ is converged, i.e. $U(x_{m,n}) > U_G(x^{(nb)}_{m,n})$, we update the solution stored in the current neighbour node $x^{(nb)}_{m,n}$ with $U_G(x^{(nb)}_{m,n})$, and update it in the nodes $x^{(nb)}_{m^*,n^*}$ that coincide with the current one.

7. Then we check if $x^{(nb)}_{m,n}$ and its coincide nodes $x^{(nb)}_{m^*,n^*}$ are presented in the active list L, i.e. if (m,n) and (m^*,n^*) are found in L. If not, we add them to the active list L before the node with indices (i,j).

8. After we finish parsing and updating all neighbours, we remove the current node $x_{i,j}$ from the active list L and continue iterative computation until L is empty.

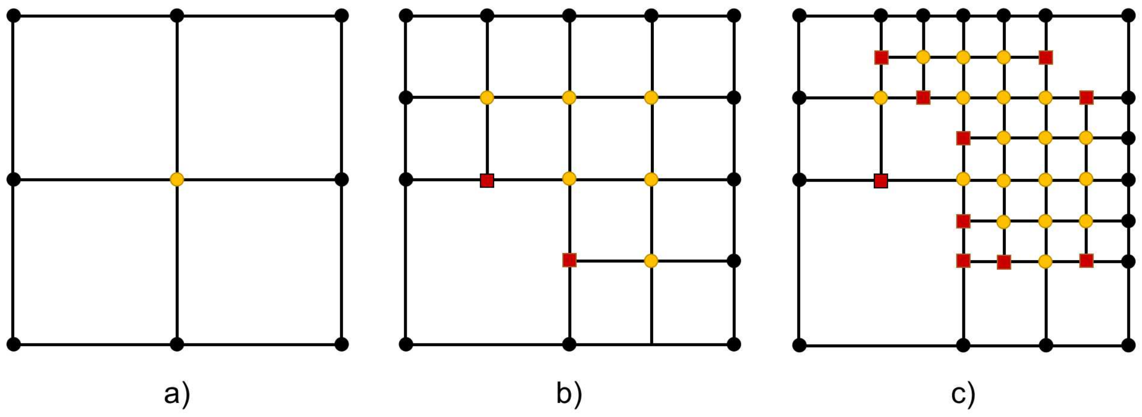


Figure 4.5: *T-mesh example obtained using a quadtree subdivision of space with boundary (black dots), crossing (yellow dots) and T-junction (red quads) nodes; a) the initial mesh, b) mesh after the first refinement, c) mesh after the second refinement.*

9. Finally, we construct the PHT-spline interpolating surface using the computed basis functions, basis vertices and Beźier coefficients in each cell and restore at least C^1 continuous solution of the eikonal equation. The computed unsigned distance field can be seen in Fig. 4.3 (c).

In the next subsections we will cover the main steps of the algorithm in more details.

4.2.3 Hierarchical Subdivision of the Domain

To adaptively subdivide $\mathbb{R}^2$ space with a defined FRep object we need to construct a 2D T-mesh. A T-mesh is a hierarchical partition of the domain $X \in \mathbb{R}^n$ using rectangular grid that allows T-junctions. T-junctions appear during the hierarchical subdivision of space where the difference between levels of subdivisions is greater than one (see Fig. 4.5, red quads). A particular case of the T-mesh is a quadtree/octree data-structures that subdivide the domain X in quad cells or cubes of different size according to the level of details. A quadtree recursively subdivides the space into four nodes, whilst octree subdivides the space into eight nodes at each iteration. We show a typical quadtree in Fig. 4.4 (a), and its tree pointer-based node structure in Fig. 4.4 (b).

We denote that a grid point in a quadtree is called a node of the quadtree. The node that belong to the boundary of the domain we call a boundary node (see Fig. 4.5, black circles). The vertex that belong to the interior of the T-mesh and belong to the centre of the crossings is called a crossing node (see Fig. 4.5, yellow circles). The node that forms a T-junction is called T-junctional (see Fig. 4.5, red quads).

4.2.4 Discretisation Scheme

In this work we consider the numerical solution of the eikonal equation $||\nabla\phi(\boldsymbol{p})|| = 1$ ($\boldsymbol{p}$ is a point in $\mathbb{R}^3$) as a special case of nonlinear Hamilton-Jacobi partial differential equation (PDE), defined on a non-graded adaptive Cartesian grids with a scalar speed function. We are going to numerically solve a hyperbolic partial PDE given

Algorithm 5: Hierarchical fast iterative method

```
   Input: Function $F_{FRep}(\boldsymbol{p})$ computed on the regular grid, hierarchical grid
          $H_G$ with $i^{th}$ leaf cell nodes $x_{i,j}$ and $j^{th}$ cell corner vertices, solution
          $U(x)$, active list $L$
   Output: $U(x_{i,j})$
 1 Active list $l_k$ contains elements $(i, j)$: {
 2 $i$ - index pointing in the quadtree leaves array;
 3 $j$ - index pointing in the leaf corner vertices array $j \in [0, 3]$;
 4 };
   /* Initialisation                                                    */
 5 for all leaf nodes $x_i \in H_G$ do
 6     for $j = 0; j < 3; j++$ do
 7         if $0 \leq F_{FRep}(x_{i,j}) \leq \epsilon$ then
 8             $U(x_{i,j}) \leftarrow 0$;
 9             add $(i, j)$ to $L$;
10         else
11             $U(x_{i,j}) \leftarrow \infty$;
   /* Updating nodes in active list L                                   */
12 while $L$ is not empty do
13     for all nodes $l_k \in L$ do
14         $i = l_k.x; j = l_k.y$;
15         $U_p \leftarrow U(x_{i,j})$;
16         $U_q \leftarrow$ solution $U_G$ of $f_{Godunov}(x_{i,j}) = 0$ on $H_G$;
           /* Check if the solution $U_G(x_{i,j})$ is converged            */
17         if $U_p > U_q$ then
18             $U(x_{i,j}) \leftarrow U_q$;
19             Update the solution $U(x_{i^*,j^*}) \leftarrow U_q$ stored in all nodes $x_{i^*,j^*}$ that
                coincide with $x_{i,j}$;
20         else
21             Find neighbour nodes $x^{(nb)}_{m,n}$ of $x_{i,j}$ in the horizontal and vertical
                directions and return a pair $((m, n), U(x^{(nb)}_{m,n}))$;
               /* Check nodes $x_{nb_{m,n}}$ if they need further updates;   */
22             for all found neighbour nodes $x^{(nb)}_{m,n}$ of $x_{i,j}$ do
23                 if $U(x^{(nb)}_{m,n}) > U(x_{i,j})$ and $(m, n) \notin L$ then
24                     $U_p \leftarrow U(x^{(nb)}_{m,n})$;
25                     $U_q \leftarrow$ solution $U_G$ of $f_{Godunov}(x^{(nb)}_{m,n}) = 0$ on the $H_G$;
26                     if $U_p > U_q$ then
27                         $U(x^{(nb)}_{m,n}) \leftarrow U_q$;
28                         Update the solution $U(x^{(nb)}_{m^*,n^*}) \leftarrow U_q$ stored in all
                            nodes $x^{(nb)}_{m^*,n^*}$ that coincide with $x^{(nb)}_{m,n}$;
29                         if $(m^*, n^*) \notin L$ then
30                             add $(m^*, n^*)$ to $L$ before $(i, j)$;
31                         add $(m, n)$ to $L$ before $(i, j)$;
32             Erase $(i, j)$ from $L$;
33 Restore the solution of the eikonal equation using PHT-splines;
```

as follows (Jeong and Whitaker 2008):

$$H(\boldsymbol{p}, \nabla\phi) = |\nabla\phi(\boldsymbol{p})|^2 - \frac{1}{f_{sp}(\boldsymbol{p})} = 0 \quad \forall \boldsymbol{p} \in \Omega \subset \mathbb{R}^n \tag{4.2}$$

where $H(\boldsymbol{p}, \nabla\phi)$ is the Hamiltonian, Ω is a domain in Euclidean space $\mathbb{R}^n$, $\phi(\boldsymbol{p})$ is a travel-time or distance from the specified source seed or seeds, and $f_{sp}(\boldsymbol{p})$ is a positively defined scalar speed function. In geometric modelling speed function $f_{sp}(\boldsymbol{p})$ is equal to one.

The solution of this equation defines the propagation of the wave-front from the seed sources. The motion of the wave-front is controlled by the speed function $f_{sp}(\boldsymbol{p})$. The solution of this equation (4.2) defines the geodesic distance of the shortest path from the nearest seed point.

Typical hierarchical grid consists from cells of different sizes that results in a different step size along the grid directions. To solve the PDE equation (4.2) on the adapted hierarchical grid we use first order Godunov upwind discretisation scheme. In 2D case it can be written as follows:

$$g(\boldsymbol{p}) = \left(\frac{U(\boldsymbol{p}) - U_x(\boldsymbol{p})}{h_x}\right)^2 + \left(\frac{U(\boldsymbol{p}) - U_y(\boldsymbol{p})}{h_y}\right)^2 - \frac{1}{f_{sp}^2(\boldsymbol{p})} = 0 \tag{4.3}$$

We need to solve it analytically for $U(\boldsymbol{p})$ to obtain a numerical scheme that is suitable for computing on non-graded adaptive grids. The solution can be obtained in a straight-forward manner:

$$U(\boldsymbol{p}) = \left(\frac{U_x}{h_x^2} + \frac{U_y}{h_y^2} \pm \frac{\sqrt{h_x^2 + h_y^2 - f_{sp}^2(\boldsymbol{p})(U_x(\boldsymbol{p}) - U_y(\boldsymbol{p})^2}}{f_{sp}(\boldsymbol{p})h_x h_y}\right)\frac{h_x^2 h_y^2}{h_x^2 + h_y^2} \tag{4.4}$$

and can be easily generalised for the 3D case. Among two solutions we chose the one that is positive. This type of update is called two-sided update as both neighbours $U_x(\boldsymbol{p})$ and $U_y(\boldsymbol{p})$ are taken into account during computations. The solution is accepted only when the following condition is satisfied: $U(\boldsymbol{p}) \geq \max(U_x(\boldsymbol{p}), U_y(\boldsymbol{p}))$. This is the upwind condition introduced in (Gómez et al. 2019) that is rewritten here for the case of the adapted grids as follows:

$$|U_x(\boldsymbol{p}) - U_y(\boldsymbol{p})| \leq \frac{\min(h_x, h_y)}{f_{sp}(\boldsymbol{p})} \tag{4.5}$$

If the following upwind condition failed, then the one sided update is computed:

$$U(\boldsymbol{p}) = \min(U_x(\boldsymbol{p}), U_y(\boldsymbol{p})) + \frac{\min(h_x, h_y)}{f_{sp}(\boldsymbol{p})} \tag{4.6}$$

Let us formulate the Algorithm 6 for implementing the first order upwind Godunov discretisation scheme that was used in the introduced HFIM Algorithm 5. The input for this algorithm is a node $x_{i,j}$ for which we are going to compute a new solution. Here index i points at the quadtree leaves array, and index j points at the corner vertices array, where we store the solution. The main steps of the algorithm are as follows:

1. First, we need to find all neighbour leaves of the current leaf x_i.

Algorithm 6: First order Godunov discretisation scheme for adaptive grids

```
Input: the node $x_{i,j}$. Here we assume that $f_{sp} = 1$.
Output: $U(x_{i,j})$
1 Find all neighbour nodes $x_{m,n}^{(nb)}$ for $x_{i,j}$ in horizontal and vertical directions
    taking into account T-junctions;
2 Get stored solutions and steps in neighbour nodes along the grid as tuples
    $(U_R(x_{m,n}^{(nb,R)}), h_R), (U_L(x_{m,n}^{(nb,L)}), h_L), (U_T(x_{m,n}^{(nb,T)}), h_T)$ and $(U_B(x_{m,n}^{(nb,B)}), h_B)$;
  /* Compare two solutions in horizontal and vertical directions and
     return the minimum one with corresponding step;                     */
3 $(U_X, h_X) = \min((U_R(x_{m,n}^{(nb,R)}), h_R), (U_L(x_{m,n}^{(nb,L)}), h_L))$;
4 $(U_Y, h_Y) = \min((U_T(x_{m,n}^{(nb,T)}), h_T), (U_B(x_{m,n}^{(nb,B)}), h_B)$;
  /* Find minimum solution between $U_X$ and $U_Y$ and return a tuple with
     corresponding step;                                                 */
5 $(U_{min}, h_{min}) = \min((U_X, h_X), (U_Y, h_Y))$;
  /* Here $f_{sp}$ is a speed function that we choose equal to one;       */
6 if $|U_X - U_Y| \leq h_{min}/f_{sp}$ then
7     $U(x_{i,j}) = \left( \frac{U_X}{h_X^2} + \frac{U_Y}{h_Y^2} \pm \frac{\sqrt{h_X^2 + h_Y^2 - f_{sp}^2 (U_X - U_Y)^2}}{f_{sp} h_X h_Y} \right) \frac{h_x^2 h_y^2}{h_x^2 + h_y^2}$;
8 else
9     $U(x_{i,j}) = U_{min} + \frac{h_{min}}{f_{sp}}$;
```

2. Then we process the found leaves and search for the neighbour nodes of the current node $x_{i,j}$ in horizontal and vertical directions. If in one of the directions we get a T-junction node, we need to process it and restore the 'ghost' value by applying the interpolation operation defined by equation (4.7), which was introduced in subsection 4.2.5. Here and further, by 'ghost' value we will assume a non-existing node opposite the T-junctional node (see Fig. 4.5 imaginary neighbour nodes opposite to red quads). In our implementation the output of this step is a vector of tuples with currently stored solution $U(x_{m,n}^{(nb)})$ and step h along the grid.

3. After we have obtained all tuples $(U_R(x_{m,n}^{(nb,R)}), h_R)$, $(U_L(x_{m,n}^{(nb,L)}), h_L)$, $(U_T(x_{m,n}^{(nb,T)}), h_T)$ and $(U_B(x_{m,n}^{(nb,B)}), h_B)$ with stored solutions and steps, we need to find the tuples (U_X, h_X) and (U_Y, h_Y) with minimum solutions in horizontal and vertical directions with the corresponding steps.

4. To compute the upwind condition defined by the equation (4.5), we need to find the minimum solution between U_X and U_Y and return corresponding tuple (U_{min}, h_{min});

5. Then, if the upwind condition holds true, we compute the two sided update defined by the equation (4.4). Otherwise, we compute a one sided update defined by the equation (4.6) using the values stored in tuple (U_{min}, h_{min}) as we want to minimise the computing solution.

4.2.5 Treatment of T-Junctions

To apply Godunov discretisation scheme we need to compute first derivatives of the function $U(\boldsymbol{p})$ on the non-regular Cartesian grid. This can be done by applying

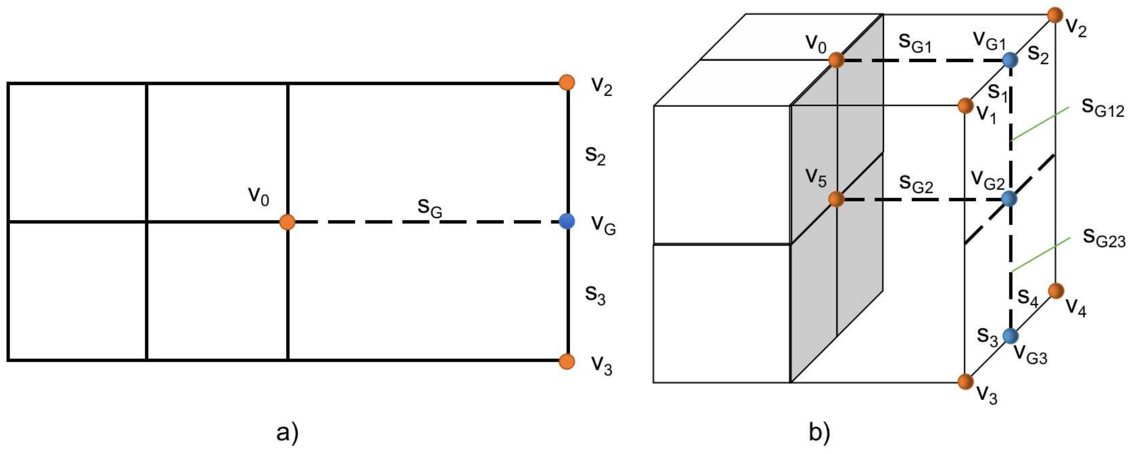


Figure 4.6: *A scheme for computing 'ghost' values in case of the T-junctions presence in the obtained hierarchical grid.*

finite difference discretisation of the first derivatives. The most problematic points for computing this discretisation are T-junction nodes, i.e. a node for which there is a missing neighbour node in one of the Cartesian directions. In Fig. 4.6 (a) we show the typical case of the T-junction in 2D, where the node ν_0 is a T-junction node, ν_G is a ghost neighbouring node with its two neighbour nodes ν_2 and ν_3. In this work we suggest to compute the value of the node-sampled function $U(\boldsymbol{p})$ in the ghost node ν_G as follows:

$$U(\nu_G) = \frac{U(\nu_2)s_3 + U(\nu_3)s_2}{s_2 + s_3} \tag{4.7}$$

Similar interpolation scheme can be obtained for 3D case. Here as it follows from Fig. 4.6 (b) there are two possible cases when we need to compute ghost values. One case corresponds to the nodes ν_{G_1}, ν_{G_3} and another to the node ν_{G_2}. To compute the value of the node-sampled function $U(\boldsymbol{p})$ in the ghost node ν_{G_1}, we can use a second-order linear interpolation:

$$U(\nu_{G_1}) = \frac{U(\nu_2)s_1 + U(\nu_1)s_2}{s_1 + s_2} \tag{4.8}$$

To compute the value of the node-sampled function $U(\boldsymbol{p})$ in the ghost node ν_{G_2}, we can use a bilinear interpolation:

$$U(\nu_{G_2}) = \frac{s_{G_{23}}s_2U(\nu_1) + s_{G_{23}}s_1U(\nu_2) + s_{G_{12}}s_2U(\nu_3) + s_{G_{12}}s_1U(\nu_4)}{(s_{G_{12}} + s_{G_{23}})(s_1 + s_2)} \tag{4.9}$$

4.2.6 PHT-Spline Interpolation

As it follows from the HFIM algorithm, introduced in subsection 4.2.2, we need to subdivide the Euclidean space according to the FRep function $F_{FRep}(\boldsymbol{p})$ values and restore the distance field at each cell using PHT-splines.

A PHT-spline can be classified as a parametric representation (see subsection 2.2.1). It is a piecewise bicubic polynomial that is defined over hierarchical grids. It can be also considered as a generalisation of B-splines (Deng et al. 2008) over hierarchical grids.

According to the subsection 4.2.3, a hierarchical grid can be constructed using a quadtree. While subdividing the domain using quadtree, we need to compute Bézier coefficients, basis functions and basis indices for further construction of the PHT-spline interpolation. This can be done, for example, using an adaptation of the NURBS model fitting algorithm (Deng et al. 2008). After applying these steps we obtain a PHT-spline interpolating surface that can be further used for restoring the distance field at each cell of the T-mesh defined by the quadtree.

Spline spaces over T-meshes

Formally, a linear spline space $L(m, n, \beta, \alpha, T)$ over T-mesh T (Deng et al. 2008) can be defined as:

$$L(m, n, \beta, \alpha, T) := \{f(x, y) \in C^{\alpha,\beta}(X) : f(x, y)|_\tau \in \mathbb{P}_{m,n}, \forall\tau \in T\} \tag{4.10}$$

where $C^{\alpha,\beta}(X)$ is the continuity space of all bivariate functions in X that have C^α continuity along X-axis and C^β continuity along Y-axis, $\mathbb{P}_{m,n}$ is the set of all tensor product polynomials with bi-degree (m, n), and τ is the cell of the T-mesh T. In this work we consider the following C^1 continuous spline space $L(3, 3, 1, 1, T)$ with a dimension defined as follows:

$$\dim L(3, 3, 1, 1, T) = 4(N_b + N_+); \quad m \geq 2\alpha + 1; \quad n \geq 2\beta + 1 \tag{4.11}$$

where N_b is the number of boundary vertices and N_+ is the number of crossing vertices.

Blossoming with De Casteljau's Algorithm

Here we briefly describe how the blossoming with De Casteljau's Algorithm 7 is done following the logic of the work (Anitescu, Hossain, and Rabczuk 2018).

To compute the Bézier representation of the basis functions $b(\xi_1, \xi_2)$ on the T-mesh nodes, we need to subdivide the parent node $\tau_{l,j}$ into four sub-nodes $\tau_{l+1,j_1}, \tau_{l+1,j_2}, \tau_{l+1,j_3}$ and τ_{l+1,j_4}. To evaluate m Bézier polynomials $b_i^{(0)} \in [\xi_k, \xi_{k+1}]$ in each spatial direction and to subdivide the node $\tau_{l,j}$ into four sub-nodes, we apply De Casteljau's algorithm (Piegl and Tiller 1997) that in one dimension can be written as follows:

$$b_i^{(j)}(\xi_0) = (1 - \xi_0)b_i^{(j-1)}(\xi_0) + \xi_0 b_{i+1}^{(j-1)}(\xi_0) \quad i = 1, ..., m - j + 1; \quad j = 1, ..., m \tag{4.12}$$

where $\xi_0 = 0.5$ is a fixed knot ξ and its equality to 0.5 is essential for obtaining a symmetric polynomial subdivision. This algorithm produces two sets of Bézier polynomials $b_0^{(0)}, b_0^{(1)}, ..., b_0^{(m)}$ and $b_0^{(m)}, b_1^{(m-1)}, ..., b_m^{(0)}$ that are defined on two new segments $[\xi_k, (\xi_k + \xi_{k+1})/2]$ and $[(\xi_k + \xi_{k+1})/2, \xi_{k+1}]$.

In two dimensions, when we are processing a quadtree hierarchical data-structure, we compute $(m + 1)^2$ Beźier coefficients $C_{i,j}^{(\tau)}$ corresponding to a single basis function that is stored in a $(m + 1) \times (m + 1)$ matrix. We also need to compute Beźier coefficients for four children nodes $\tau_{ch} \equiv \tau_{l,j_k}, k = 1, ..., 4$ that are stored in a $2(m + 1) \times 2(m + 1)$ matrix. First, we compute $m + 1$ times 1D De Casteljau's algorithm for each row of the matrix $C_{i,j}^{(\tau_m)}$ and then $2(m+1)$ times for each column of the resulting $(m + 1) \times 2(m + 1)$ array. Here τ_m corresponds to a parent node that we are going to split with 2D De Casteljau's Algorithm 8.

Algorithm 7: De Casteljau's algorithm in 1D

```
Input: Beźier coefficients $b_1^{(0)}, ..., b_{m+1}^{(0)}$
Output: $b_i^{(j)}(0.5)$
/* Split the input curve defined with Beźier coefficients
   $b_1^{(0)}, ..., b_{m+1}^{(0)}$ into two curves at $\xi_0 = 0.5$          */
1 for $j = 1; j \leq m; j++$ do
2     for $i = 1; i \leq m - j + 1; i++$ do
3         $b_i^{(j)}(0.5) = (b_i^{(j-1)}(0.5) + b_{i+1}^{(j-1)}(0.5))/2$;
```

Algorithm 8: De Casteljau's algorithm in 2D

```
Input: $C_{i,j}^{(\tau_m)}$ for the parent node, polynomial degrees $m$ and $n$
Output: new $C_{i,j}^{(\tau_m)}$
/* Split the input Bézier coefficients $C_{i,j}^{(\tau_m)}$ matrix into four
   matrices corresponding to the new child nodes $\tau_{ch}$                */
1 for $int\ i = 0; i < C_{i,j}^{(\tau)}.rows(); i++$ do
      /* Apply 1D De Casteljau's algorithm in the row direction            */
2     for $j = 0; j < n + 1; j++$ do
3         $[C_{split,1}, C_{split,2}]$= computeDeCasteljau_1D($C_{i,j}^{(\tau_m)}$);
4         $C_{j,all}^{(\tau_{ch})} = [C_{split,1}, C_{split,2}]$;
      /* Apply 1D De Casteljau's algorithm in the column direction         */
5     for $j = 0; j < 2(m + 1); j++$ do
6         $[C_{split,1}, C_{split,2}]$= computeDeCasteljau_1D($C_{0,..,n,j}^{(\tau_{ch})}$);
7         $C_{all,j}^{(\tau_{ch})} = [C_{split,1}, C_{split,2}]$;
```

Truncation of Beźier Coefficients

The construction algorithm for the hierarchical basis functions has two problems. The first of them is the violation of the unity partition:

$$\sum_{i=0}^{m} B_{i,m}(\xi^{(1)}, \xi^{(2)}) \neq 1 \quad 0 \leq \xi^{(1)}, \xi^{(2)} \leq 1$$

Another one is the increase in the number of overlapping basis functions associated with different hierarchical levels. Both problems can be solved by introduction of a truncation operation for the hierarchical basis functions (Anitescu, Hossain, and Rabczuk 2018; Kiss, Giannelli, and Jüttler 2014). This operation provides the linear independence of the constructing basis functions and a sparse resulting linear system.

Here we follow the approach described in (Anitescu, Hossain, and Rabczuk 2018). The constructed basis functions of a polynomial degree m are assumed to be at least $C^{\alpha,\alpha}$ continuous with the following condition $2\alpha + 1 \leq m$ that holds true. In our case we consider the generated PHT-spline surface to be at least $C^{(1,1)}$ continuous.

The truncation process starts from splitting each of the four child nodes $\tau_{l,j}$ into nine regions: four corner regions, four edge regions and one centre region (see Fig. 4.7). Then, as soon as we have obtained the new basis vertex, we have to set to zero all Beźier coefficients in the regions that are close to it, i.e. one corner, two edge and centre regions.

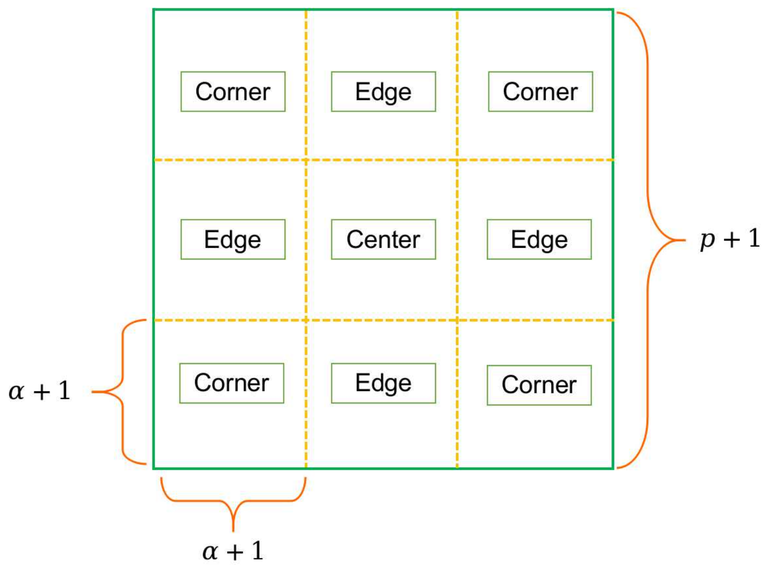


Figure 4.7: *The T-mesh nodes $\tau_{l,j}$ subdivided into nine regions with specified number of Beźier ordinates in each sub-region. The total number of Beźier ordinates is $(m+1)\times(m+1)$ and each sub-region contains $(\alpha+1)\times(\alpha+1)$ ordinates.*

Adding New Basis Functions

The refinement procedure is finished by insertion of the new basis functions in cells $\tau_{l,j}$ of the constructed T-mesh. The insertion of the new basis functions (Anitescu, Hossain, and Rabczuk 2018) consists of several steps. First, we have to compute corresponding Beźier ordinates $C_{i,j}^{(\tau)}$ and new global basis indices. The new basis functions form non-truncated B-splines that are computed using local knot vector information. The local knot vector is defined for each new basis vertex. It consists of three distinct knots that include the parametric coordinates of the knot-span endpoints to the left and right of the new basis vertex as well as to the up and down of it. In general case, the continuity $C^{\alpha,\alpha}$ of the constructed B-spline depends on the multiplicity of the knots. The multiplicity for interior knots is $m-\alpha$ and for boundary knots is $m+1$. As we are considering splines with at least $C^{1,1}$ continuity, the multiplicity for interior knots will be $m-1, \alpha=1$.

During the refinement process of the T-mesh that includes cross-insertion operation, some basis functions are set to zero according to the truncation step. Then, new basis functions are added to the nodes $\tau_{l,j}$ after cross-insertions. The new basis indices that are assigned to the new basis functions can be either the basis indices corresponding to the removed basis functions or incrementally generated new ones. If the polynomial degree is $m=2\alpha+1$, then the construction algorithm for computing new basis functions is simplified and follows the method described in (Jeong and Whitaker 2008).

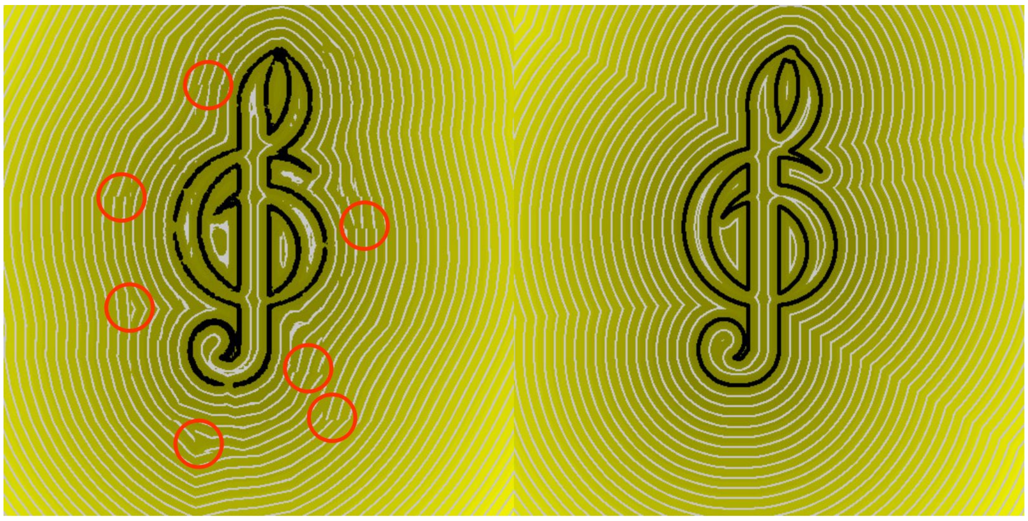


Figure 4.8: *The comparison of the field restoration at each subdivided hierarchical cell using bilinear interpolation (a) and PHT-spline interpolation (b). In red circles we can see the C^0 discontinuity in the field isolines where the cells of different size appear next to each other.*

PHT-Spline Surface

The tensor product PHT-spline surface $S(\xi^{(1)}, \xi^{(2)})$ for the spline space $L(3,3,1,1,T)$ is defined using the Bézier representation (Deng et al. 2008):

$$S(\xi^{(1)}, \xi^{(2)}) = \sum_{i=1}^{m}\sum_{j=1}^{m} C_{i,j}^{(PHT)} P_{i,j}(\xi^{(1)}, \xi^{(2)}) \tag{4.13}$$

$$P_{i,j}(\xi^{(1)}, \xi^{(2)}) = \sum_{i=1}^{m+1}\sum_{j=1}^{m+1} C_{i,j}^{(\tau)} B_{i,j}(\xi^{(1)}, \xi^{(2)}) \tag{4.14}$$

where $m = 3$ is the polynomial degree, $(\xi^{(1)}, \xi^{(2)}) \in [0,1] \times [0,1]$ are knot vectors, $C_{i,j}^{(PHT)}$ are the control points of the constructing PHT-spline surface $S(\xi^{(1)}, \xi^{(2)})$, $P_{i,j}(\xi^{(1)}, \xi^{(2)})$ are the PHT-basis functions in the Beźier form. Here $C_{i,j}^{(\tau)}$ are the scalar Bézier ordinates of the basis functions computed for each child node $\tau_{l+1,j}$ of the T-mesh. $B_{i,j}(\xi^{(1)}, \xi^{(2)})$ are the Bernstein basis functions that are defined as a tensor product of Bernstein polynomials $B_{i,j}(\xi^{(1)}, \xi^{(2)}) = B_i(\xi^{(1)}) \otimes B_j(\xi^{(2)})$(Piegl and Tiller 1997).

As we have stated in subsection 3.2.3, the ADF field has C^0 discontinuities that arise after the hierarchical subdivision where cells of different size appear. In Fig. 4.8 (a) the discontinuities in the white isolines are located in the red circles. C^1 discontinuities are introduced by the bilinear/trilinear interpolation that is used for the field restoration in interior of each cell (see Fig. 4.8, a). As it can be seen in Fig. 4.8 (b) the field generated by our method with PHT-spline restoration of the distance field successfully solves these drawbacks. All the isolines are continuous and smooth.

4.3 Generation of IDF

In this subsection we consider the step two of the basic algorithm (subsection 3.6) with respect to generating IDF. IDFs are usually computed using the solution of

Algorithm 9: HFRep based on FRep and IDF

```
Input: the FRep field $F_{FRep}(\boldsymbol{p})$, fixed source point $\boldsymbol{p}_{src}$, the HFRep field $F_{HFRep}(\boldsymbol{p})$
Output: $|F_{HFRep}(\boldsymbol{p}_{src}, \boldsymbol{V}_l^*)|$
1  Extracting FRep boundary: set of vertices and edges $(V, E)$ in 2D or set of vertices and faces $(V, F)$ in 3D;
2  Embed the extracted FRep surface $\mathbb{R}^n \mapsto \mathbb{R}^m$ using diffusion map: $\boldsymbol{V}_i \mapsto (e^{-\lambda_1 t}\phi_1(\boldsymbol{V}_1), ..., e^{-\lambda_n t}\phi_n(\boldsymbol{V}_i))$;
3  Obtain eigenvectors $\phi_n(\boldsymbol{V}_i))$ and eigenvalues $\lambda_n$ of the computed Laplace-Beltrami operator on the boundary isocontour or surface;
   /* Compute pairwise distances on the embedded surface: */
4  for $i = 0; i < size\_of(V); i++$ do
5      for $j = 0; j¡size\_of(V); j++$ do
6          $F^2_{DF}(\boldsymbol{V}_i, \boldsymbol{V}_j) = \sum_{k=1}^{n} e^{-2\lambda_k t}(\phi_k(\boldsymbol{V}_i) - \phi_k(\boldsymbol{V}_j))^2$;
7  Triangulate 2D isocontour or tetrahedralise generated FRep surface and obtain new vertices $\boldsymbol{V}_l^*$;
   /* Compute diffusion distances between source point $\boldsymbol{p}_{src}$ and vertices $\boldsymbol{V}_l^*$ using barycentric interpolation: */
8  for $l = 0; l < size\_of(V^*); l++$ do
9      for $i = 0; i < size\_of(V); i++$ do
10         for $j = 0; j < size\_of(V); j++$ do
11             $|F_{HFRep}(\boldsymbol{p}_{src}, \boldsymbol{V}_l^*)| = F^2_{IDF}(\boldsymbol{p}_{src}, \boldsymbol{V}_l^*) = \sum_{i,j} F^2_{DF}(\boldsymbol{V}_i, \boldsymbol{V}_j)\omega_i(\boldsymbol{p}_{src})\omega_j(\boldsymbol{V}_l^*) - -\frac{1}{2}\sum_{i,j} F^2_{DF}(\boldsymbol{V}_i, \boldsymbol{V}_j)(\omega_i(\boldsymbol{p}_{src})\omega_j(\boldsymbol{p}_{src}) + \omega_i(\boldsymbol{V}_l^*)\omega_j(\boldsymbol{V}_l^*))$;
12 if Generate signed HFRep == true then
13     Voxelise obtained mesh;
14     Extrapolate distances in exterior of the mesh and get $F^*_{IDF}(\boldsymbol{p}_l)$;
       /* Generate HFRep on voxel grid $G_{vox}$ with points $\boldsymbol{p}_q$: */
15     for $q = 0; q < size\_of(G_{vox}); q++$ do
16         $F_{HFRep}(\boldsymbol{p}_q) = (F_{st} \circ F_{FRep})(\boldsymbol{p}_q) \cdot F^*_{IDF}(\boldsymbol{p}_q)$;
```

some PDE equations or, alternatively, some graph-based approach. The generation of IDFs is based on propagation of the distances computed on the boundary of the mesh in its interior. We suggest to use the method described in (Rustamov, Lipman, and Funkhouser 2009). We briefly outline the theoretical aspects of this method that are essential for its reproduction.

4.3.1 Theoretical Background

According to the work (Rustamov, Lipman, and Funkhouser 2009), the computation of IDF consists of three steps. Let us assume that we have a triangular mesh with vertices $\boldsymbol{\nu}_i \in \mathbb{R}^3, i = 0, ..., n$. First, we embed these vertices $\boldsymbol{\nu}_i$ in some m-dimensional $\mathbb{R}^m$ space using a map $\boldsymbol{\nu}_i \mapsto \boldsymbol{\nu}_i^* \in \mathbb{R}^m$. This map was suggested to compute using diffusion maps introduced in (Coifman and Lafon 2006). It can be obtained by computing an eigendecomposition $\{\lambda_k, \phi_k\}_{k=1}^n$ of a discrete Laplace-

Beltrami operator of the mesh. Then the diffusion map can be written as

$$\boldsymbol{\nu}_i \mapsto (e^{-\lambda_1 t}\phi_1(\boldsymbol{\nu}_1), ..., e^{-\lambda_n t}\phi_n(\boldsymbol{\nu}_i)) \tag{4.15}$$

where t is the time parameter, λ_i is the i^{th} eigenvalue and ϕ_i is the i^{th} eigenvector of the Laplace-Beltrami operator. Then the diffusion distance (Goes, Goldenstein, and Velho 2008) can be written as a pairwise Euclidean distance:

$$F_{DF}^2(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j) = \sum_{k=1}^{n} e^{-2\lambda_k t}(\phi_k(\boldsymbol{\nu}_i) - \phi_k(\boldsymbol{\nu}_j))^2 \tag{4.16}$$

After we have computed diffusion distances on the surface of the mesh, we need to extend them to the interior of the mesh using barycentric interpolation. If point $\boldsymbol{p} \in G_{in}$, then the barycentric representation of it can be defined as:

$$\boldsymbol{p} \mapsto \boldsymbol{p}^* = \sum_i \omega_i(\boldsymbol{p})\boldsymbol{\nu}_i \tag{4.17}$$

where $\omega_i(\cdot)$ are barycentric coordinates (e.g. mean-value coordinates).

Finally, distances in interior of the mesh can be obtained using computed diffusion distances $F_{DF}(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j)$ and barycentric interpolation as follows:

$$\begin{aligned} F_{IDF}^2(\boldsymbol{p}, \boldsymbol{q}) = &\sum_{i,j} F_{DF}^2(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j)\omega_i(\boldsymbol{p})\omega_j(\boldsymbol{q}) - \\ &- \frac{1}{2}\sum_{i,j} F_{DF}^2(\boldsymbol{\nu}_i, \boldsymbol{\nu}_j)(\omega_i(\boldsymbol{p})\omega_j(\boldsymbol{p}) + \omega_i(\boldsymbol{q})\omega_j(\boldsymbol{q})) \end{aligned} \tag{4.18}$$

To compute the barycentric interpolation we used mean-value coordinates that were defined in 2D case according to (Hormann and Floater 2006) and in 3D according to (Ju, Schaefer, and Warren 2005).

4.3.2 Interior Distance Field Generation

To construct an HFRep based on FRep and IDF, we need to do several steps according to the Algorithm 9. First, we need to extract the boundary of the FRep object O_{FRep} and use obtained points $\boldsymbol{\nu}_i$ for the computation of the diffusion map defined by equation (4.15) and, then, diffusion distances according to the equation (4.16). In Fig. 4.9 we show the generated IDF field for the 'heart' object. In Fig. 4.9 (a) we can see the generated diffusion map for the current object.

Then, we need to tetrahedralise the obtained mesh. After the diffusion distances were computed on the surface of the mesh as it can be seen in Fig. 4.9 (b), they are extended to the interior of the tetrahedralised mesh according to equation (4.18) using barycentric interpolation with, e.g. mean-value coordinates. In Fig. 4.9 (c) we show the obtained distances in the interior of the 'heart' object.

Finally, if we need to obtain distances in the exterior of the object, we convert the tetrahedral mesh to the voxel representation preserving the computed IDFs. Thereafter, we can apply an extrapolation operation (e.g., using a wavenumber based extrapolation (Tam and Kurbatskii 2000)) to the obtained IDF field to propagate distances to exterior of the object. This operation will allow us to make the IDF field signed at the last step of the basic algorithm. This step has not been implemented yet and it is a matter of the future work.

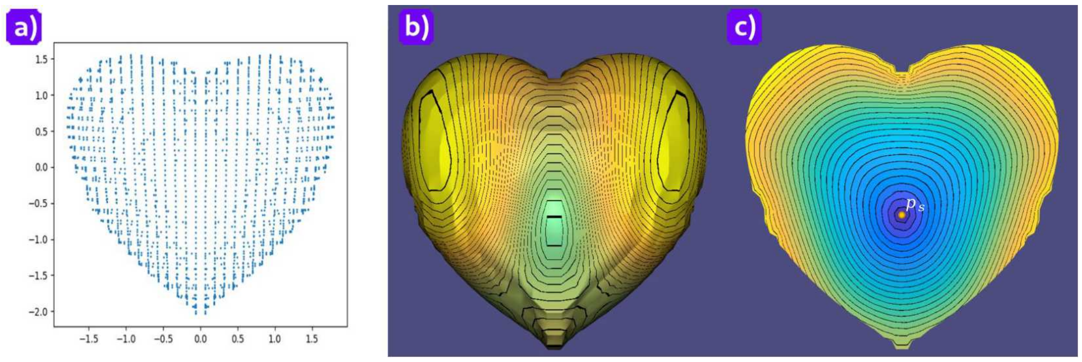


Figure 4.9: *HFRep based on hybridisation of FRep and IDF representations. This hybridisation was applied to a FRep 'heart' object. The method for generating the IDF representation was taken from (Rustamov, Lipman, and Funkhouser 2009). a) The diffusion map that was computed on the triangulated surface of the FRep 'heart' object that is used for restoring distances on the boundary of the shape; b) The distances obtained on the boundary of the object shape that are shown as black isolines; c) The tetrahedral slice of the mesh with isolines corresponding to the interior distances. The yellow point $\boldsymbol{p}_s$ corresponds to the 'source' point defined in the object interior.*

4.4 Smoothing an Obtained Distance Field

In this section we briefly consider the step three of the basic algorithm (section 3.6). The obtained UDF function $F_{DF}(\mathbf{p})$ defined by equation (3.44) is discrete and neither smooth nor continuous as the field obtained with this function is computed in a finite number of points on a regular grid. To enforce at least C^1 continuity and essential smoothness for the obtained field, we need to use some at least C^1 continuous interpolation function. This is achieved using spline functions of degree m, such as *B-splines* or bicubic/tricubic splines (Knott 2000). Note, that the interpolation function should not introduce additional zeroes in the HFRep field. More specifically, we suggest to use a bicubic spline or B-spline interpolation for the 2D case, and a tricubic spline or B-spline interpolation for the 3D case.

4.5 Distinguishing Between Interior, Boundary and Exterior of the Object

In this section we consider the step four of the basic algorithm (section 3.6). To distinguish between interior and exterior of the HFRep object O_{HFRep}, we need to define a sign for the obtained UDF function $F_{DF}(\boldsymbol{p})$, which should be positive inside, equals zero on the border of the object O_{HFRep} and negative outside the object O_{HFRep}.

We suggest to obtain the sign using a smooth step-function that depends on the values of the FRep function defined at the first step of the basic algorithm. The step-function $F_{st}(F_{FRep}(\boldsymbol{p}))$ should satisfy the following requirements:

1. It should be approximately equal -1 when it corresponds to the exterior of the FRep object, $F_{FRep}(\boldsymbol{p}) < 0$;

2. It should be approximately equal to 0 on the boundary of the FRep object, $F_{FRep}(\boldsymbol{p}) = 0$;

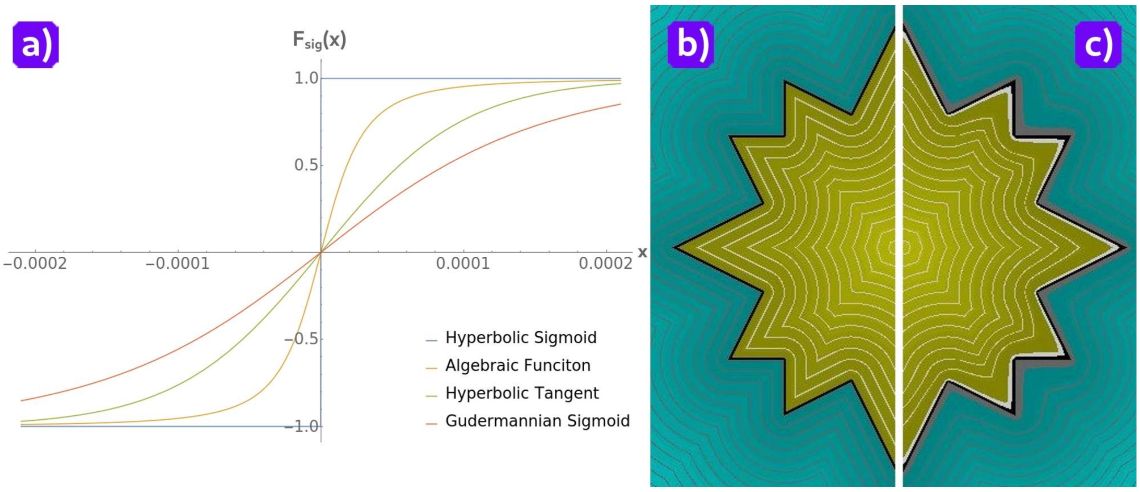


Figure 4.10: *The illustration of the HFRep function continuity through varying the slope controlling parameter s_l of the step-function. a) Plots of four sigmoid functions $F_{sig}(\cdot)$: hyperbolic tangent sigmoid function ($s_l = 10^{-10}$, blue line), algebraic function ($s_l = 10^{-9}$, orange line), hyperbolic tangent function ($s_l = 10^4$, green line) and Gudermannian sigmoid function ($s_l = 10^4$, red line). b) the HFRep 'star' object that was computed with $s_l = 0.00001$ for F_{sig}, equation (4.19); c) the HFRep 'star' object that was computed with $s_l = 0.1$ for the F_{sig}, equation (4.19); all sharp features are smooth, i.e. the HFRep function is C^1 continuous.*

3. It should be approximately equal to 1 inside the FRep object, $F_{FRep}(\boldsymbol{p}) > 0$;
4. It should be at least C^1 continuous everywhere in a Euclidean space $\mathbb{R}^n$;
5. It should barely modify the values of UDF.
6. It should not produce additional zeroes in the computed UDF field.

We have identified two classes of functions which satisfy these requirements. They are *sigmoid functions*, $F_{sig}(x), x \in \mathbb{R}$, and *spline functions*, $F_{sp}(x), x \in \mathbb{R}$. In this work we have tested four different *sigmoid functions*.

First function we have tested is the *hyperbolic tangent sigmoid function* (see Fig. 4.10, blue line). By controlling its slope parameter s_l it is possible to get nearly step-function behaviour around zero:

$$S_{sig}(x) = \frac{r}{1 + exp(-2x/s_l)} - \frac{r}{2} \tag{4.19}$$

where r controls the range of the $S(x)$ along y-axes. Parameter r should be equal to two to make the function defined by the equation (4.19) propagate from -1 to 1 along the y-axis. To obtain a step-function like behaviour, the slope parameter should be equal to some small value, e.g. $s_l = 10^{-40}$.

The second function we have tested is *hyperbolic tangent function* (see Fig. 4.10, green line) defined as

$$S_{sig}(x) = \tanh(s_l \cdot x) \tag{4.20}$$

where s_l parameter controls the slope of the function. Its behaviour is similar to the step-function if s_l coefficient is set up as a relatively huge value, for example, 10^{40}.

Alternatively, the fourth function we have tried is an *algebraic function* (see Fig. 4.10, orange line) that is defined as:

$$S_{sig}(x) = \frac{x}{\sqrt{s_l + x^2}} \tag{4.21}$$

where s_l parameter controls the slope of the function. The s_l coefficient should be close to zero value, e.g. $s_l = 10^{-40}$.

The last function we have tested is *Gudermannian sigmoid function* (see Fig. 4.10, red line) (W. Olver et al. 2010). In the general case, it is defined as the solution of the integral of the hyperbolic secant

$$\begin{aligned} S_{sig}(x) = \int_0^x a \cdot \operatorname{sech}(s_l \cdot x)dx = a \cdot \arcsin(\tanh(s_l \cdot x)) = \\ = a \cdot \arctan(\sinh(s_l \cdot x)) = 2a \cdot \arctan(\tanh(0.5 \cdot s_l \cdot x)), \\ -\infty < x < +\infty \end{aligned} \tag{4.22}$$

If $a = 1$ the defined function is bounded along Y-axis from -1.5 to 1.5. To clip it to the new region along Y-axis from -1 to 1 we need to obtain a new value for a, which is approximately equal to 0.642129. To obtain a step-function like behaviour the slope parameter s_l should be equal to some huge value, e.g. $s_l = 10^{40}$.

4.6 HFRep: Implementation and Results

In this section we discuss our implementation of the introduced framework. The described theoretical framework is implemented as a *C++ HFRep library* of objects and methods for generating and processing such geometric objects that satisfy all theoretical aspects of the proposed framework. In particular, this implementation provides the means for dealing with HFRep objects attributes that are parameterised by the distance values of at least C^1 continuous HFRep function. The library also provides functionality for generating time-variant heterogeneous HFRep objects both in 2D and 3D. Finally, we describe our implementation of HFRep that is based on FRep and SDF representations in *SideFX Houdini* (Side Effects Software 2020). All examples in this and the following sections of this chapter were computed on a laptop with a 2.6 GHz *Intel Skylake 6700* processor and 16 Gb of RAM.

The implemented *HFRep Library* for heterogeneous volumetric modelling provides methods and tools for generating various HFRep objects both in 2D and 3D. It includes implemented methods for the generation of FRep objects, computation of UDFs for these FRep objects that depend on the chosen bi-pair hybridisation, generation of HFRep objects, assignment of attributes to them and rendering the result. To render 3D HFRep objects we have implemented a sphere-tracing technique that was discussed in section 2.6.

The *HFRep library* is implemented as a *C++* library of classes and functions, that is based on several external libraries including *OpenGl* (Silicon Graphics 2020), *OpenMP* (OpenMP Architecture Review Board 2020), *Eigen* (Jacob and Guennebaud 2020), *GTS* (*The GNU Triangulated Surface Library* 2020) and LibIGL (Jacobson, Panozzo, et al. 2018). The implementation of the library is available on the GitHub .

4.6.1 Examples of Attribute Handling

In this subsection we show how we practically work with HFRep heterogeneous objects in terms of their attributes in the implemented *C++* HFRep library.

In subsection 3.6.2, we have outlined the basic algorithm for generating HFRep attribute functions. However, there is no universal approach for dealing with an

`https://github.com/teshaTe/HybridFrep`

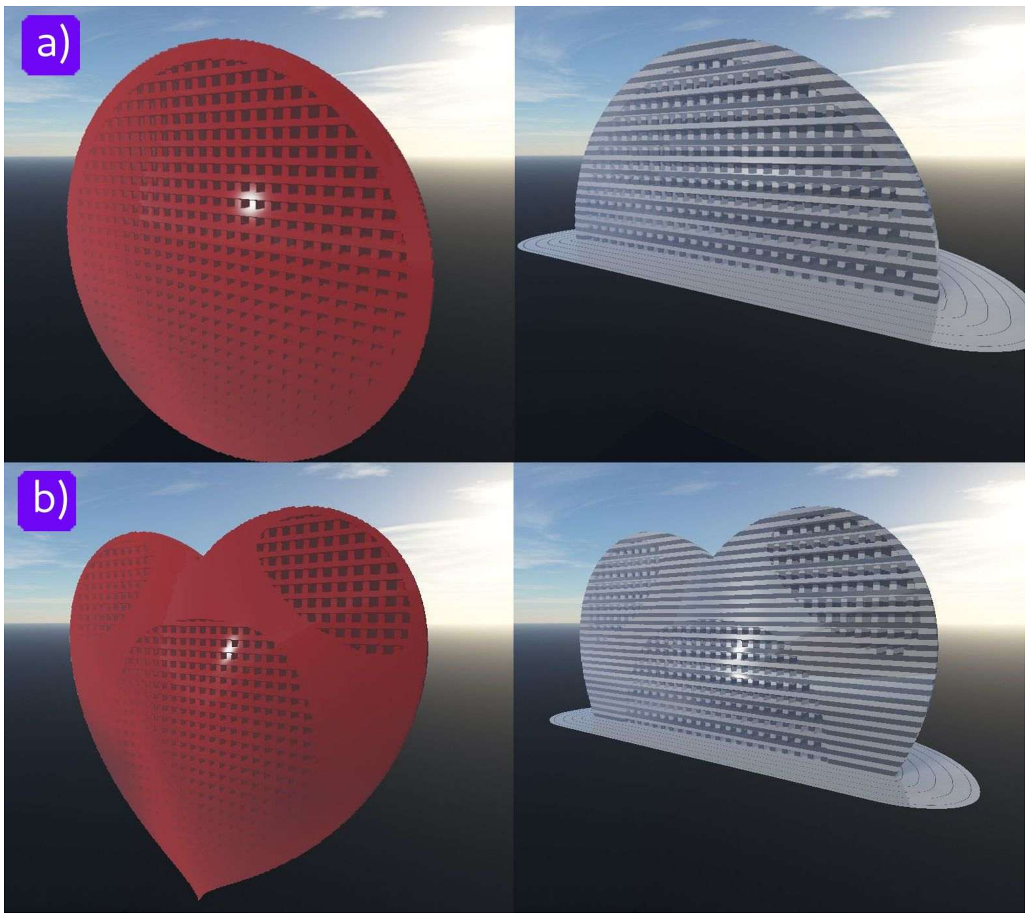


Figure 4.11: *The illustration of the HFRep heterogeneous object based on the FRep and SDF representations with incorporated microstructure. a) the rendered HFRep 'sphere' object using the sphere-tracing method (left) and its isolines (right); b) the rendered HFRep 'heart' object using sphere-tracing method (left) and its isolines (right);*

HFRep object attributes because of their various nature. In this subsection we show how the proposed framework works for some representative attributes, namely, microstructures, colour and material attributes. We discuss the following examples: the HFRep microstructures (Fig. 4.11), two heterogeneous objects with defined procedural marble material parameterised by the HFRep distance (Fig. 4.14), an HFRep 2D 'H' object with three different parameterisations of the procedural wood function by the IDF distance (Fig. 4.13) and a 2D example of metamorphosis between two HFRep objects (Fig. 4.12).

Microstructures

In Fig. 4.11 we demonstrate how microstructures in interior of the O_{HFRep} object are integrated. The microstructures were defined as incorporated infinite slabs in interior of the 'sphere' and 'heart' objects using set-theoretic operations defined by equations (3.59). The infinite slabs were defined according to (Pasko, Fryazinov, et al. 2011) as follows:

$$S(\boldsymbol{p}) = \sin(\boldsymbol{\nu} \odot \boldsymbol{p} + \boldsymbol{\phi}) + \boldsymbol{l}; \tag{4.23}$$

where $S(\boldsymbol{p}) \geq 0$ is a vector function, with components defined as a set of slabs orthogonal to either X or Y or Z-axes, $\boldsymbol{\nu}$ is a frequency vector with components

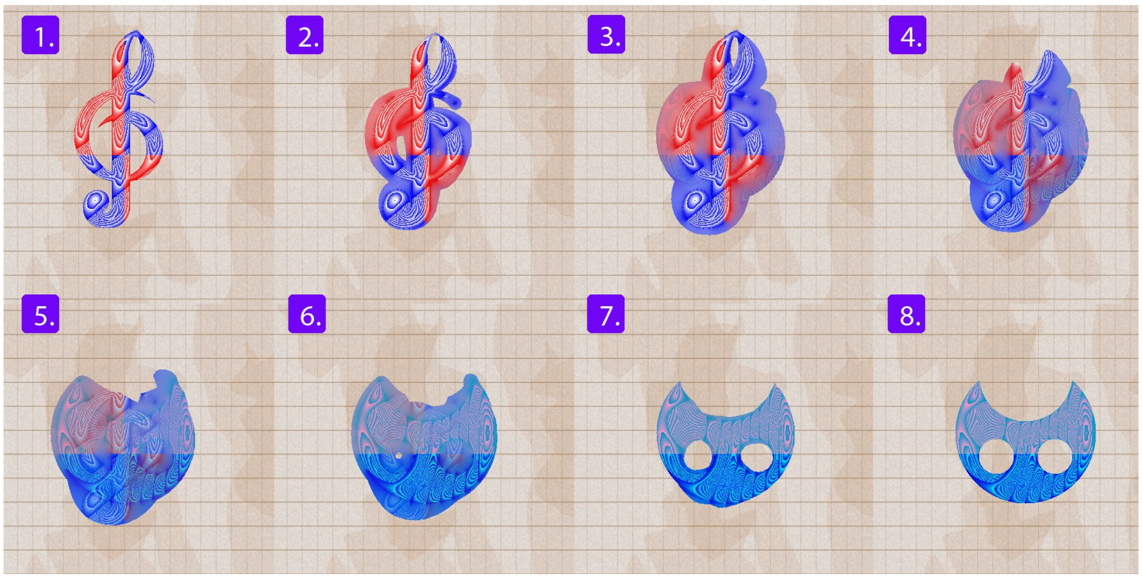


Figure 4.12: *The illustration of the metamorphosis between two HFRep textured objects using the space-time blending and space-time transfinite interpolation techniques. The texturing was made using procedural noise functions. Supplementary video: figure4.12.mpg*

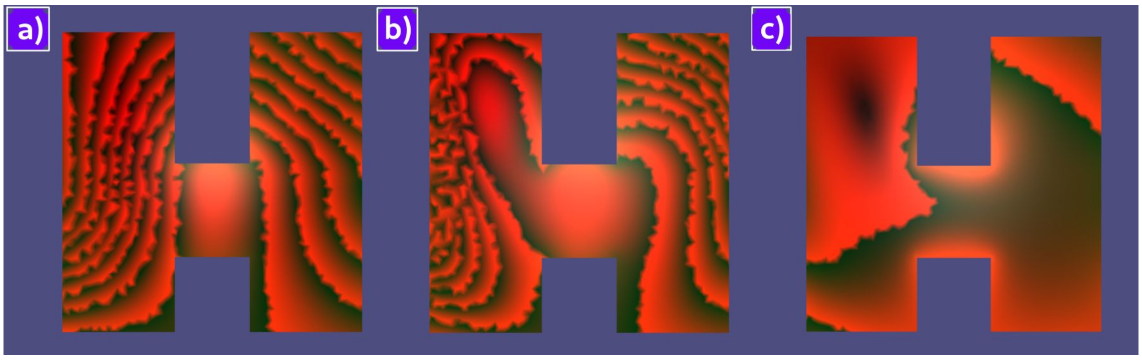


Figure 4.13: *The HFRep 'H' object that is textured using a procedural function (4.24) of the 'wood' material. This function was differently parameterised (a), (b), (c) by computed IDF for the given object.*

defined as the distance between parallel slabs along one of the axes, $\boldsymbol{p}$ is a point $\boldsymbol{p} \in X$, $\boldsymbol{\phi}$ is a phase vector with components defined as the position of slabs on one of the axes with respect to the origin, and $\boldsymbol{l}$, $-1 < l_i < 1$ is a threshold vector that together with frequency parameter controls the thickness of each slab. Then the basic algorithm was applied to the obtained function to compute the HFRep objects with microstructures.

Procedural Textures

We can specify attributes as simple procedural textures. Fig. 4.12 shows two heterogeneous HFRep objects $O_{H_v,HFRep}$ with coloured wooden textures that were obtained using a procedural function $f_{wood}(\boldsymbol{p})$. This function is constructed using hash table $htab(\boldsymbol{p})$ allowing for random sampling of the position values $\boldsymbol{p}$ multiplied by

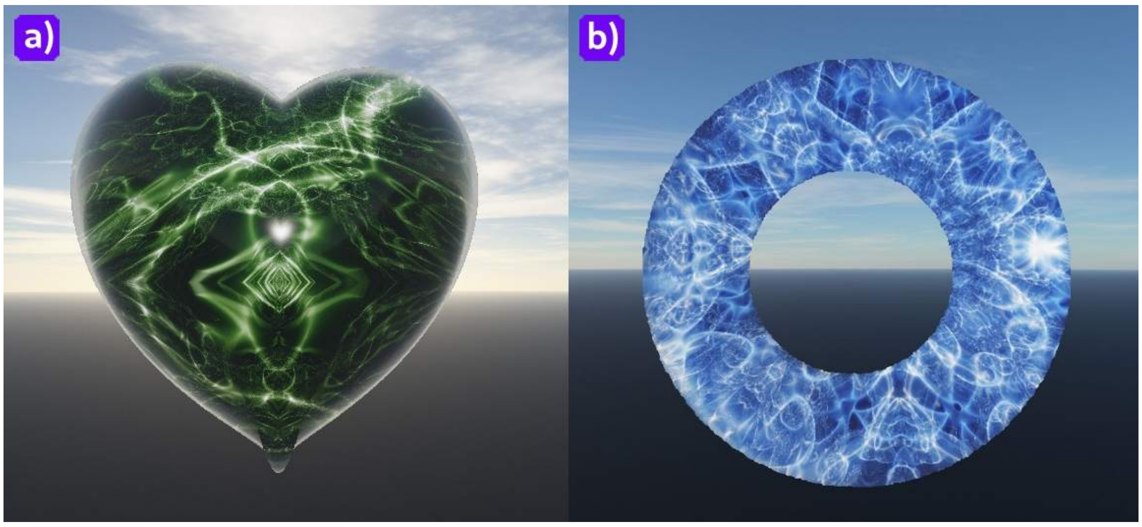


Figure 4.14: *Two heterogeneous HFRep objects a) 'heart' and b) torus with differently parameterised and coloured marble materials.*

the frequency ν. The procedural function for the 'wood' can be defined as follows:

$$\begin{aligned} g(\boldsymbol{p}) &= htab(\boldsymbol{p} \cdot \nu) \cdot c; \\ f_{wood}(\boldsymbol{p}) &= g(\boldsymbol{p}) - int(g(\boldsymbol{p})); \end{aligned} \tag{4.24}$$

where $c > 1$ is a constant, $g(\boldsymbol{p})$ is a noise function, $int(g(\boldsymbol{p}))$ is an integer part of the function $g(\boldsymbol{p})$ output value. To parameterise $f_{wood}(\boldsymbol{p})$ by the distance, we assign the distance values to the frequency parameter ν.

Then a simple segmentation of the geometric shape of the objects was done (see Fig. 4.12 (1)). We split the shape into four regions and assign colours using the obtained HFRep distance function $F_{HFRep}(\boldsymbol{p})$ and procedural function $f_{wood}(\boldsymbol{p})$ that defines the texture of the wood. The generated objects were used as inputs for 2D heterogeneous metamorphosis on the basis of the space-time blending (STB) (Pasko, Pasko, and Kunii 2005) method to handle geometry transformation and the space-time transfinite interpolation (STTI) (Fryazinov, Sanchez, and Pasko 2015) to handle colour transformation (Tereshin, Adzhiev, Fryazinov, Marrington-Reeve, et al. 2020). This example was implemented using C++ and OpenCV. The implementation of the method is available on the GitHub .

In another Fig. 4.13 we show three textured 'H' HFRep objects. The textures for these objects were generated using three different parameterisations of the procedural function for the 'wood' by the computed IDFs. In Fig. 4.14 we show two 3D HFRep objects 'heart', (a), and 'torus', (b), with procedurally defined marble material with different parameterisation by the HFRep distance.

4.6.2 Houdini Implementation: HFRep Based on FRep and SDF

SideFX Houdini (Side Effects Software 2020) is the commercial software that covers a diverse amount of areas of 3D production, from procedural geometric modelling to animation and visual effects. It is an open environment and supports different scripting APIs that can be used through socket communication. *SideFX Houdini*

https://github.com/teshaTe/2D-metamorphosis provides native support for *Python* scripting language as well as for its internal scripting language *VEX*. *SideFX Houdini* has tools for volumes processing that can be defined using integrated *OpenVDB* voxel library (Museth 2013) or its native format for representing such type of the data. Both volume implementations support SDF generation that is important for constructing our hybrid framework. As *SideFX Houdini* is widely used in the industry, provides tools for the volumetric modelling with SDF generation and flexible with providing API, we decided to implement our hybrid framework that is based on FRep and SDF inside it.

Geometry Representation

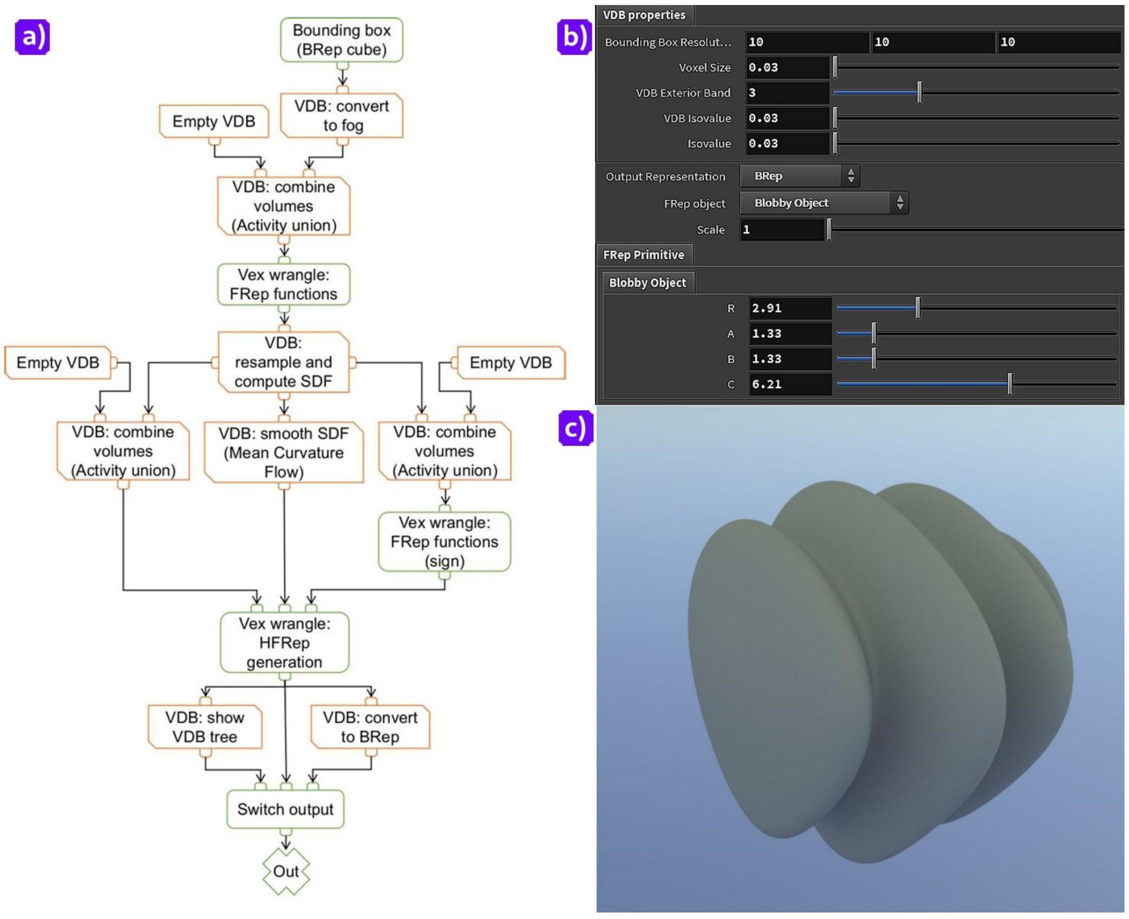
The modelling pipeline in *SideFX Houdini* consists of nodes that define various operations over different components of the system. These nodes form digital assets that are constructed by connecting different operators defined in nodes. Some of them allow the user to add new functionality to the system. Therefore, the modelling system is flexible and follows the procedural paradigm.

In this subsection, we describe how HFRep objects were implemented in *SideFX Houdini*. The procedure for the creation of HFRep objects is wrapped into a digital asset that can be further used in any project as a separate modelling node. The HFRep digital asset consists of *OpenVDB* node operators called VDB and VEX operator (VOP) nodes. In Fig. 4.15 we show how the HFRep framework is created inside *SideFX Houdini* (a), the interface of the HFRep node (b), and rendered HFRep object (c) that was defined as a blobby object.

The implementation of the HFRep node follows the basic algorithm described in section 3.6. First, as we are dealing with a VDB voxel-based representation, we need to define the bounding box that will be further voxelised for creating a computational grid. Initially, the bounding box is defined as a BRep cube that is further converted into VDB fog. As we need to obtain an empty VDB volume in the form of the bounding box, we create an empty volume VDB node and combine it with already generated fog using 'activity union' operation. Then we can use the obtained computational voxel grid for generating FRep objects.

The FRep objects are defined in *VEX Volume Wrangle VOP* using internal *SideFX Houdini* VEX scripting language. Currently, 24 primitives, namely sphere, box, cylinder, decocube, chair surface, Hunt surface, Pilz surface, heart surface, blobby, Kushner-Shmitt surface, McMullen model, quartic cylinder, tangled cube, Klein bottle, Lame surface, tear drop surface, torus, ellipsoid, orthocircle surface, Barth-Desic surface, elliptic cylinder, octohedron, pyramid and cone are implemented. The list of the primitives can be easily extended, by adding new defining equations into the *VEX Volume Wrangle VOP*.

To reduce the amount of voxels in the computational grid after we have generated a FRep object, we apply a VDB resample operation and compute SDF. It disables the unused voxels in the initial grid and fits the VDB bounding box to efficiently store the generated geometry. Then we generate a smoothed SDF using the mean curvature flow interpolation method. We also need to recompute the FRep field on the resampled VDB grid to use it for restoring the sign of the HFRep field. We define an empty VDB volume where we will store the computed HFRep field in the *VEX Volume Wrangle VOP*. All the input data is computed on the voxel grids of the same size. To restore the sign for the HFRep field we use the *sigmoid function* defined using equation (4.19). As an output we provide the VDB representation, the VDB tree data structure and the BRep representation.


a)
Bounding box (BRep cube)
Empty VDB
VDB: convert to fog
VDB: combine volumes (Activity union)
Vex wrangle: FRep functions
Empty VDB
VDB: resample and compute SDF
Empty VDB
VDB: combine volumes (Activity union)
VDB: smooth SDF (Mean Curvature Flow)
VDB: combine volumes (Activity union)
Vex wrangle: FRep functions (sign)
Vex wrangle: HFRep generation
VDB: show VDB tree
VDB: convert to BRep
Switch output
Out
b)
c)

VDB properties

Bounding Box Resolut...	10	10	10
Voxel Size	0.03		
VDB Exterior Band	3		
VDB Isovalue	0.03		
Isovalue	0.03		
Output Representation	BRep		
FRep object	Blobby Object		
Scale	1		

FRep Primitive

Blobby Object

R	2.91
A	1.33
B	1.33
C	6.21

Figure 4.15: *The SideFX Houdini implementation of the HFRep representation based on hybridisation of FRep and SDF. a) The node diagram implementation of HFRep; b) The user interface of the HFRep node; c) The rendered HFRep blobby object.*

Reinitiliasation Operation for The HFRep Field

As a user can apply various operations to the HFRep objects, the implemented framework should take into account that some of the operations (see section 3.8) preserve distance property and some of them violate it. According to the basic algorithm (see section 3.6) we need to apply reinitialiasation operation. This operation repeats all the steps of the basic algorithm and restore the distance property for the HFRep object. Let us explain how we have implemented this operation in *SideFX Houdini*.

In Fig. 4.16 we show the diagram of the reinitialisation algorithm. The input for the implemented node is the HFRep scalar field that after one or several applied operations lost its distance property. First, we need to restore the distance property and compute the SDF field. We resample and rebuild the SDF field using a VDB node. Then we need to initialise an empty VDB volume of the same size, where we will store the regenerated HFRep field. We also need to smooth the computed SDF field using the mean curvature flow interpolation method. As all fields should be defined on the voxel grids of the same size, we need to reshape the input HFRep field voxel grid according to the restored SDF voxel grid. Finally, we can regenerate HFRep distance field. This node outputs both the HFRep field and its BRep representation.

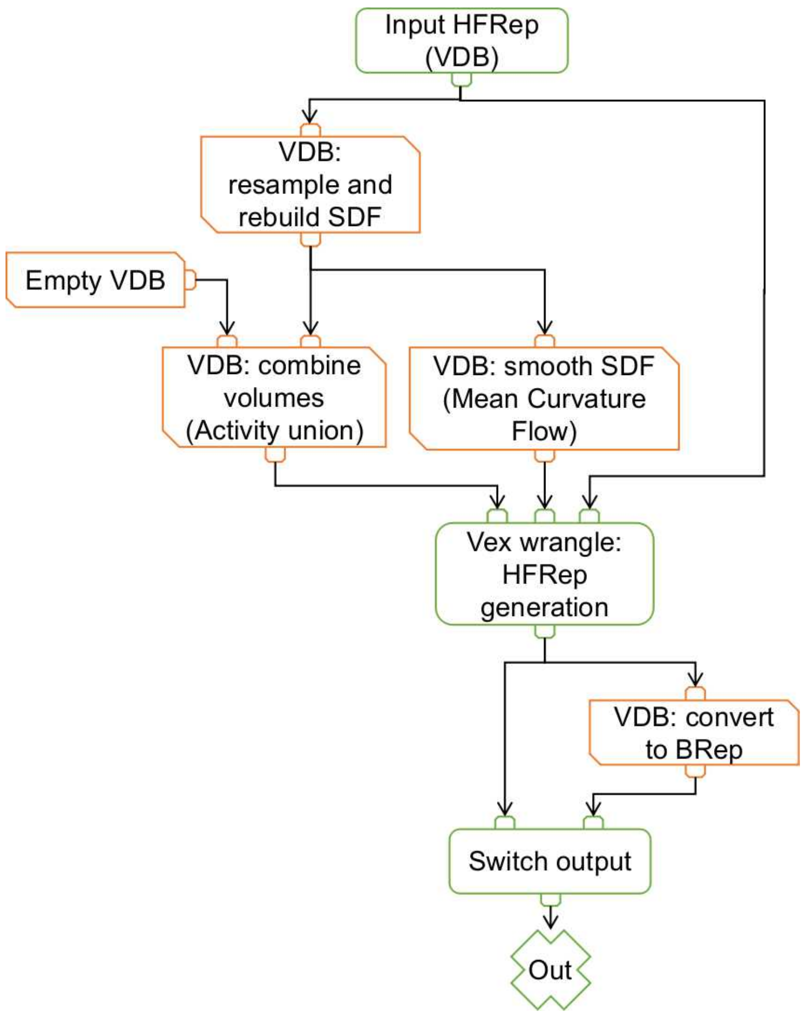

Input HFRep (VDB)
VDB: resample and rebuild SDF
Empty VDB
VDB: combine volumes (Activity union)
VDB: smooth SDF (Mean Curvature Flow)
Vex wrangle: HFRep generation
VDB: convert to BRep
Switch output
Out

Figure 4.16: *The SideFX Houdini implementation of the HFRep reinitialiasation operation.*

Implementation of the Set-theoretic Operations

Most of the operations, such as offsetting, bending, tapering and some others (see section 3.8) that are applicable to scalar fields, including SDFs, are already implemented in *SideFX Houdini*. We have implemented set-theoretic operations defined by at least C^1 continuous R-function system (3.59) and by another at least C^0 continuous R-function system (3.58). Our algorithmic implementation of these operations inside *SideFX Houdini* is shown in Fig. 4.17 (a).

This node takes as an input a pair of distance-based fields (two HFReps or two SDFs or their mix). First, we need to obtain a new VDB voxel grid with its size equal to the sum of two input voxel grids and use it for initialisation of the empty VDB volume. Then we can pass two input fields to the *VEX Volume Wrangle VOP*, where we have implemented set-theoretic operations. The output of this node can be either an HFRep or SDF field defined in the VDB voxel grid or its BRep representation. In Fig. 4.17 (b) and (c), we show the result of application of the set-theoretic intersection operation between 'Barth Desic' and 'sphere' objects using equations (3.59).

We use our implementation of the set-theoretic operations for generating shelled HFRep or SDF objects. In Fig. 4.18 (a) we show a diagram of the algorithm for producing a shelled object. In Fig. 4.18 (b) we show an initial object to which we

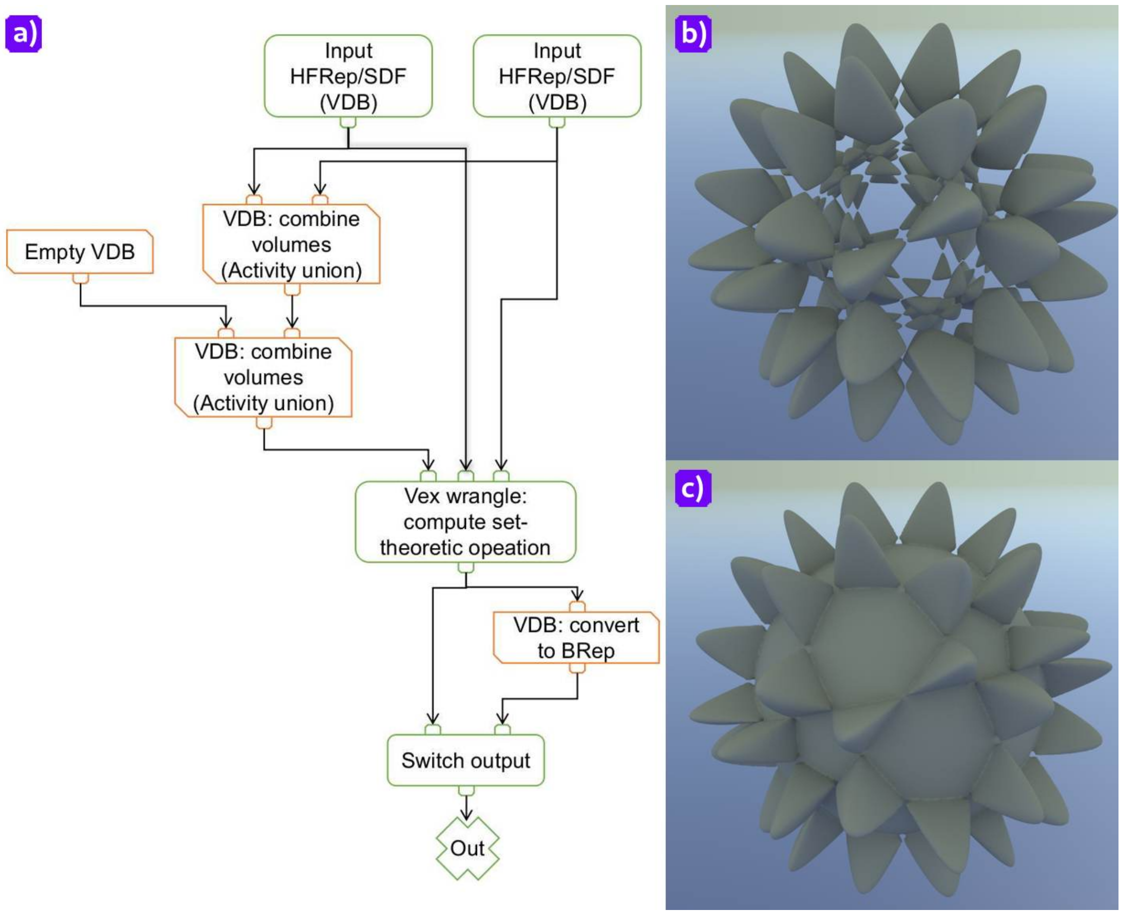

a)
Input HFRep/SDF (VDB)
Input HFRep/SDF (VDB)
Empty VDB
VDB: combine volumes (Activity union)
VDB: combine volumes (Activity union)
Vex wrangle: compute set-theoretic opeation
VDB: convert to BRep
Switch output
Out
b)
c)

Figure 4.17: *The SideFX Houdini implementation of the set-theoretic operations in the form of two R-function systems. a) The diagram of the algorithm; b) A Barth-Desic functionally defined object; c) A union of the Barth-Desic object with sphere object defined by HFRep functions.*

apply the algorithm for producing a shelled object, and, in Fig. 4.18 (c), we show the result of the operation.

This operation has one input corresponding to either an HFRep or SDF defined object stored in a VDB voxel grid. First we need to negatively offset the input distance field and convert it to VDB. Then we need to reshape an obtained VDB computational grid according to the input of the node to align voxel grid sizes before further processing of the volumetric geometry. At the next step we apply the set-theoretic operation of subtraction to remove a portion of the material from the interior of the input volumetric shape defined by either an HFRep or an SDF representation.

If we need to cut the object, we specify an empty VDB volume and define a cutting plane in *VEX Volume Wrangle VOP*. Then we can subtract the voxelised plane from the closed shape of the shelled object defined by HFRep or SDF. The output of this operation is an HFRep object defined in a VDB voxel grid or its BRep representation.

Algorithmic Solutions for Defining Various Attributes

There is a huge variety of attributes that can be specified in the interior of the heterogeneous objects. In this subsection we will describe some methods for their definition in the interior of the HFRep object that we have implemented in the

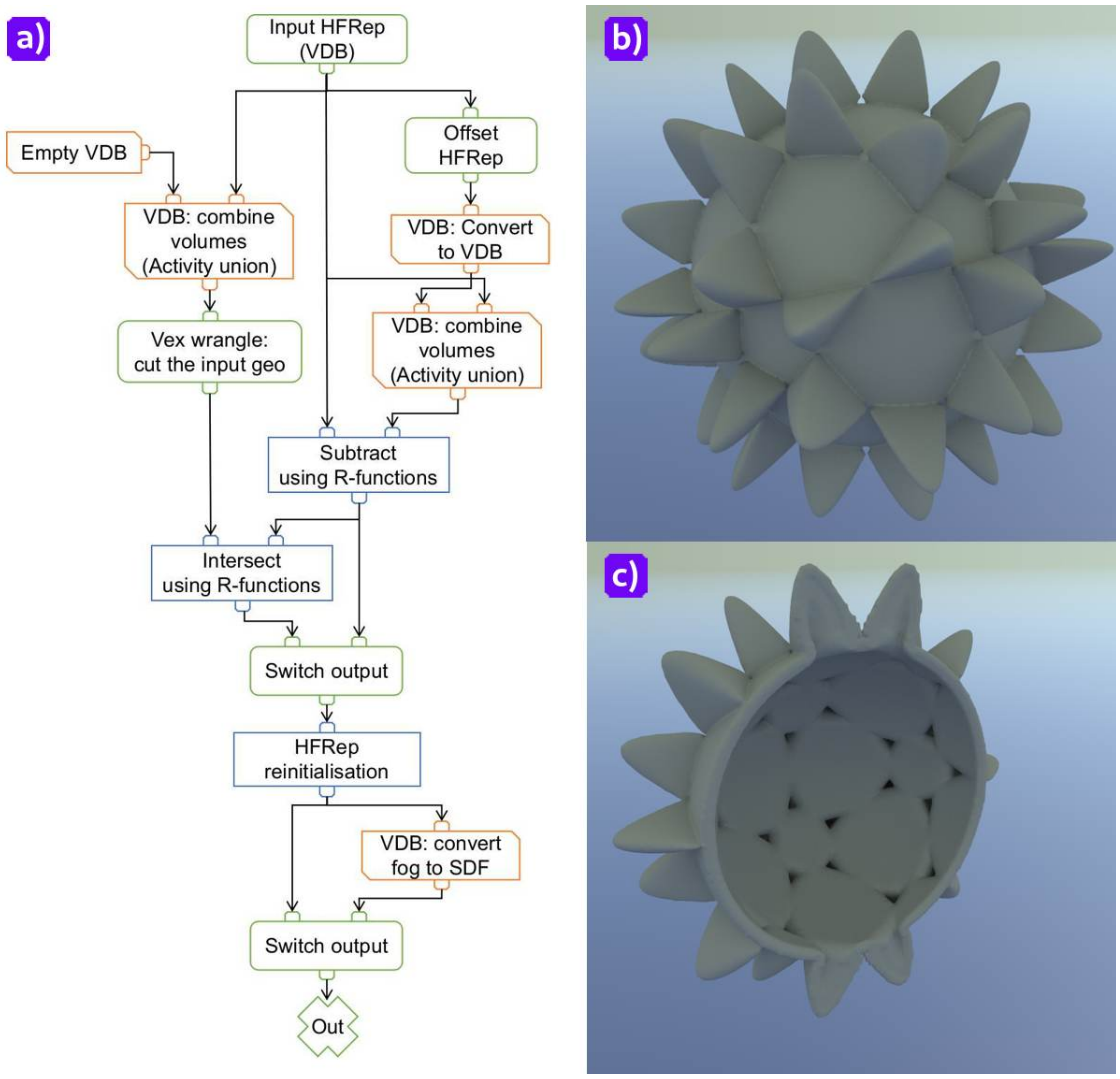
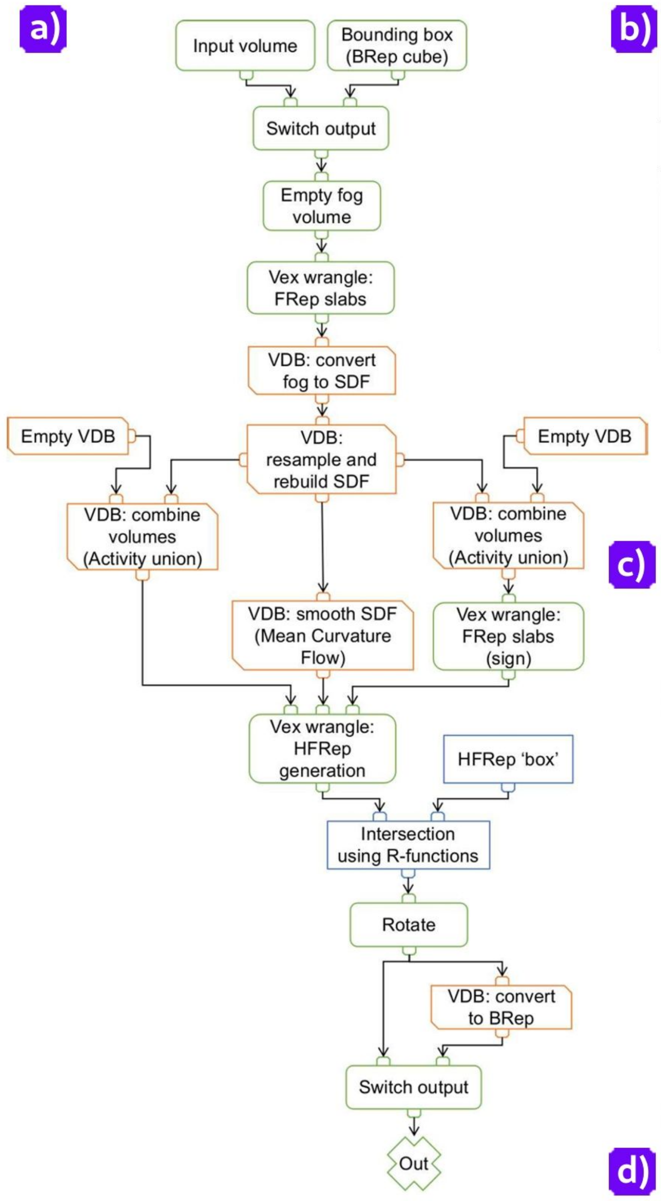

a)
Input HFRep (VDB)
Empty VDB
Offset HFRep
VDB: combine volumes (Activity union)
VDB: Convert to VDB
Vex wrangle: cut the input geo
VDB: combine volumes (Activity union)
Subtract using R-functions
Intersect using R-functions
Switch output
HFRep reinitialisation
VDB: convert fog to SDF
Switch output
Out
b)
c)

Figure 4.18: *The SideFX Houdini implementation of the shelled object. a) The diagram of the algorithm; b) A union of the Barth-Desic object with sphere object defined by the functions; c) A shelled object.*

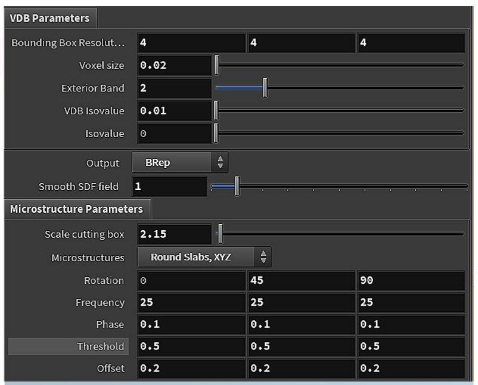
SideFX Houdini.

One of the important attributes of interior structure of practically any solid object is microstructures. The method for the definition of such objects was introduced in (Pasko, Fryazinov, et al. 2011) using a FRep representation. As the introduced hybrid framework is based on the unification of the FRep representation with distance-based representations, we can implement it and generate distance-based microstructures. The algorithmic implementation of the node for the generation of the HFRep microstructures in *SideFX Houdini* is shown in Fig. 4.19 (a). Its user interface is presented in Fig. 4.19 (b).

This node has one optional input that can be useful when the computed microstructures will be inserted in interior of the HFRep or SDF object. This optional input allows the node to use the same dimensions of the input VDB volume for computations. This is important if the generated microstructures will be further inserted in the input object. Otherwise, the size of the volumetric grid that will be used for computations can be defined manually. The volumetric grid is defined using *Volume VOP*. Then in the *VEX Volume Wrangle VOP* we define slabs with a round or squared shape using FRep functions. The output of this node will be a a)


Input volume
Bounding box (BRep cube)
Switch output
Empty fog volume
Vex wrangle: FRep slabs
VDB: convert fog to SDF
Empty VDB
VDB: resample and rebuild SDF
Empty VDB
VDB: combine volumes (Activity union)
VDB: combine volumes (Activity union)
VDB: smooth SDF (Mean Curvature Flow)
Vex wrangle: FRep slabs (sign)
Vex wrangle: HFRep generation
HFRep 'box'
Intersection using R-functions
Rotate
VDB: convert to BRep
Switch output
Out

b)

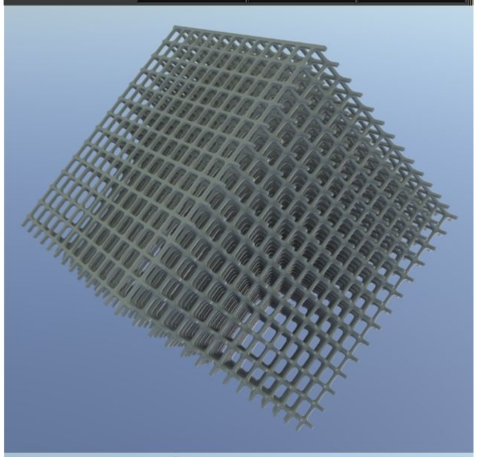

VDB Parameters

Bounding Box Resolut...	4	4	4
Voxel size	0.02		
Exterior Band	2		
VDB Isovalue	0.01		
Isovalue	0		
Output	BRep		
Smooth SDF field	1		

Microstructure Parameters

Scale cutting box	2.15		
Microstructures	Round Slabs, XYZ		
Rotation	0	45	90
Frequency	25	25	25
Phase	0.1	0.1	0.1
Threshold	0.5	0.5	0.5
Offset	0.2	0.2	0.2

c)



d)

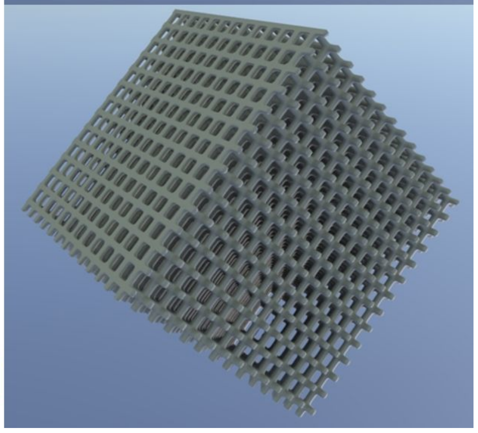


Figure 4.19: *The SideFX Houdini implementation of the HFRep microstructures. a) The diagram of the algorithm; b) The user interface of the node; c) and d) are the output of the node: slabs with round (c) and squared shapes (d).*

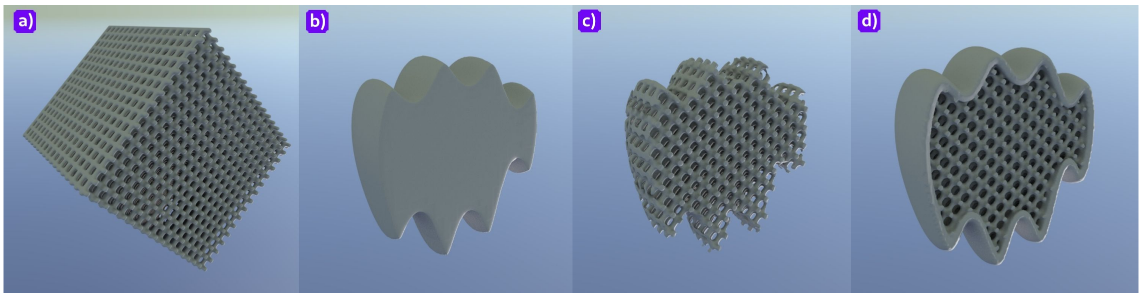


Figure 4.20: *Four stages of the microstructures incorporation inside the HFRep object that were implemented inside the SideFX Houdini.*

fog volume.

To generate the SDF field using VDB operators, we need to convert the computed fog to SDF using the VDB conversion node. Then to reduce the amount of active voxels we apply the VDB resampling operation with rebuilding of the SDF field.

Then we follow the logic of the HFRep node architecture described earlier in this subsection. We define two empty VDB volumes of the same size. One volume will be used for storing the HFRep representation of the slabs and another for computing the FRep functions of the slabs on the new VDB grid. Finally, we generate the HFRep representation of the slabs in the *VEX Volume Wrangle VOP*.

As the slabs defined by the FRep function are infinite, we need to intersect them with the HFRep cube with the size slightly less then VDB bounding box to obtain a watertight HFRep geometry. Microstructures can be rotated in space before their insertion in the geometry. The output of the current node can be either represented as a VDB volume or a BRep surface. In Fig. 4.19 (c) and (d), we show the output of the implemented node. In Fig. 4.19 (c) we show the rendered slabs with a round shape and, in Fig. 4.19 (d), we show the rendered slabs with a squared shape.

The insertion of the modelled HFRep slabs into an HFRep geometry is straightforward. It is based on the algorithms already discussed in this subsection. In Fig. 4.20, we show in a step-by-step manner the process of the microstructure insertion in the HFRep 'blobby' object. In Fig. 4.20 (a), we show the generated HFRep microstructures. Then, as we want to insert them into 'blobby' object, we need to execute three steps. First, we need to offset the 'blobby' object volume as it is shown in Fig. 4.20 (b). It will correspond to the interior of the object. At the next step, we intersect the generated offset volume with the microstructures. The result of this operation can be seen in Fig. 4.20 (c). Finally, we intersect the offset 'blobby' volume with an initial volume to obtain a shelled object and union the result with the microstructures. The result can be seen in Fig. 4.20 (d).

In Fig. 4.21 we show a 3D model of the COVID-19 cell that was obtained using 207 set-theoretic operations. In Fig. 4.21 (b), we can see the interior structure of the COVID-19 cell (Mousavizadeh and Ghasemi 2020). The central part representing the RNA and N-protein was defined using SDF that was further combined with the HFRep spherical shell of the cell. The M-protein was also defined as a combination of the SDF arc and two HFRep spheres. The rest of the elements were defined using HFRep. Each element is mono-coloured and colours are assigned per-voxel.

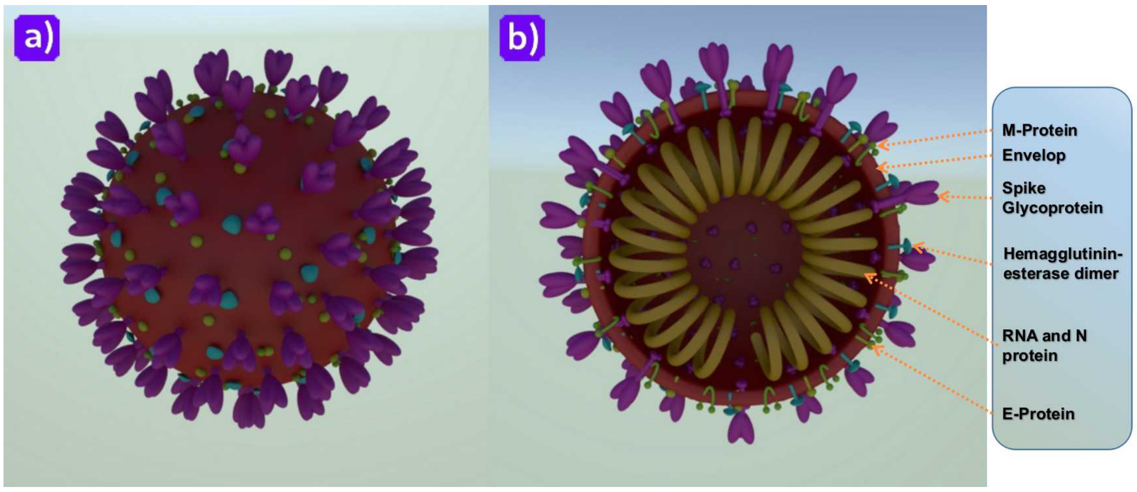


Figure 4.21: *The COVID-19 cell model obtained as a combination of the HFRep and SDF functions. a) Exterior of the virus cell; b) Interior of the virus cell.*

4.7 Conclusions

In this chapter we have provided a detailed description of the algorithmic framework and technical implementation of the basic algorithm introduced in section 3.6.

First, we have described the algorithms for generating the unsigned distance field at the second step of the basic algorithm. The distance field can be computed using one of the methods for the generation of SDF, ADF or IDF.

To generate SDFs, we have provided an algorithmic formalisation for two distance transforms, namely the signed sequential Euclidean distance transform (SSEDT) and the vector-city vector distance transform (VCVDT). We have also shown how to implement the signed VCVDT algorithm which was originally introduced as an unsigned distance transform.

To generate ADFs, we have proposed an extended FIM method for computation of the solution of the eikonal equation on hierarchical grids. We name this method as a hierarchical FIM (HFIM). We have provided a detailed description of the algorithm for the 2D case that covers the process of quadtree generation and computation of the PHT-spline interpolation. This allows us to obtain at least C^1 continuous smooth unsigned distance field.

Finally, to generate IDFs, we have used an algorithm introduced in (Rustamov, Lipman, and Funkhouser 2009). We have developed a detailed step-by-step procedure and discussed how to compute IDFs for the FRep objects and explained how to generate the signed IDFs.

Then, according to the third step of the basic algorithm, we have to smooth the generated unsigned distance field. We have given some recommendations on smoothing techniques that can be applied in the context of the proposed hybrid framework.

As the generated field is unsigned, we need to restore the sign at the fourth step. To do so, we have identified several continuous smooth-step functions that have a behaviour similar to the step functions.

Finally, we have discussed our implementation of the hybrid framework for multi-material heterogeneous modelling. To prove the concept, we have implemented several examples of the generation of heterogeneous HFRep objects. The attributes for those objects were defined as a simple per voxel colouring or procedurally (e.g. textures and microstructures). We have also presented a detailed implementation of HFRep based on FRep and SDF in *SideFX Houdini*.

In the next chapters we will show how some of the algorithms introduced in this chapter can be applied in such applications as an automatically controlled 2D metamorphosis between textured shapes and an '4D Cubism' artistic application that is extended with attributes. Both applications utilise the distance-based representations for describing 2D textured geometric shapes or heterogeneous 3D/4D objects with multi-material attributes as input. This proves that the hybrid framework introduced in the previous chapter along with its algorithmic implementation presented in this chapter can be practically used.

Chapter 5

2D Heterogeneous Space-time Blending with Automatic Control

2D imagery is a well-established area dealing with generation and transformation of both raster and vector images. Image dynamics is a more complex field concerned with time-variant image transformations. 2D shape modelling is also considered a well-researched area with its own established set of operations on shapes. However, these areas are not always well connected to each other. We will consider these matters from a perspective of a heterogeneous shape modelling and computer animation using the theoretical and practical framework developed in this work.

Morphing (or metamorphosis) means the visually smooth transition between two given images or shapes. It is a commonly used operation in computer animation, visual effects, computer art and design and it is especially relevant for consideration in the above-defined context. There are several well-known solutions for 2D image and shape metamorphosis, all of which require establishing some form of one-to-one correspondences between two given images, boundaries of two given shapes or between their particular features as it was discussed in section 2.5.

The general case where no correspondences between two given shapes or their features and no foreground image segmentation are provided has not been fully investigated yet, although several usage scenarios exist, for example:

1. In-between frames generation in a situation where there is no prior information about the source and target frames, such as in a game environment, where characters and assets are procedurally generated on the fly (Smelik et al. 2014);

2. In a live TV show or any streaming on-line content, which changes dynamically with no time available for establishing correspondences between frames;

3. In interactive applications aimed at non-professional users, or at users with specific requirements, where the input can be generic and no correspondences between source and target frames can be established.

All of the above cases are united by a lack of resources (time, information) or skills (cognitive, educational) for establishing correspondences between two given shapes or images. The important thing is that in all mentioned cases there is no need or no opportunity to establish point-to-point or feature-to-feature correspondences. It

The material in this chapter (except subsection 5.1.1 and figures 5.2, 5.15 and 5.19) was published in (Tereshin, Adzhiev, Fryazinov, Marrington-Reeve, et al. 2020). Copyright ©2020 Society for Industrial and Applied Mathematics. Reprinted with permission. All right reserved.

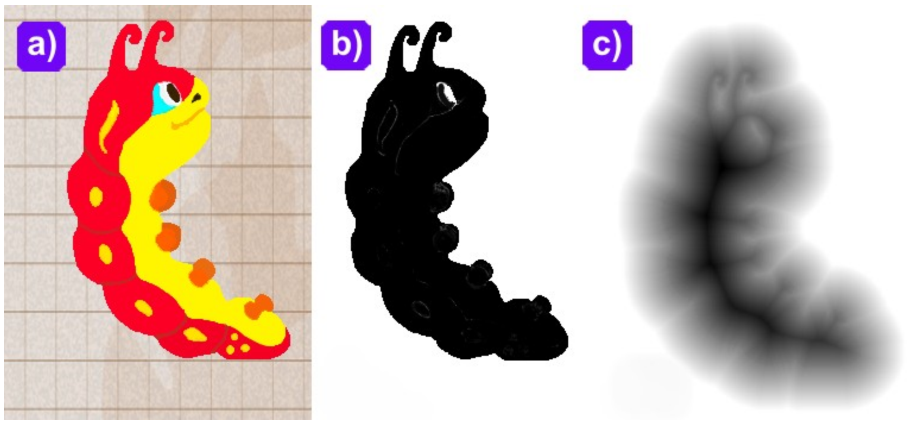


Figure 5.1: *The process of generating SDF: a) a textured shape is used as an input; b) the input shape is binarised; c) the binarised shape is used for computing the SDF.*

is made automatically and no additional in-between frames should be drawn or any parameters of the process changed by the user.

In this chapter, we present a method for computing metamorphosis of 2D shapes defined either by raster images with various textured shapes or functionally defined heterogeneous objects with defined attributes in their interior. The whole process can be described as heterogeneous object transformations, where a shape and its texture (as a 2D spatial attribute) are changing synchronously and inter-connectedly. If the initial and target objects are defined as images then they are converted to signed distance fields (SDFs). The initial and target objects with specified attributes can be also defined using distance-based functions including hybrid function representation (HFRep), SDFs, adaptively sampled distance fields (ADFs) and interior distance fields (IDFs) (only if the distances are extrapolated in exterior of the object).

Thus, to obtain morphing between two shapes with no correspondences we utilise a revised version of the metamorphosis technique called space-time blending (STB) (Pasko, Pasko, and Kunii 2005) for geometry and space-time transfinite interpolation (STTI) for colours/textures (Rvachev et al. 2001; Sanchez, Fryazinov, Adzhiev, et al. 2015). Both methods deal with functionally defined objects raised in a higher dimensional space, where the last coordinate is time. Therefore, such objects can be considered as time-dependent. Both methods do not require to establish any correspondences between input and target shapes. However, by adding constraints it is possible to obtain better shape and texture transformations.

By doing that, we allow for obtaining a smooth transition between source and target images or functionally defined objects without requirements for shape alignment, topological conformity or establishing correspondences. The method is computationally inexpensive, artist friendly and can be used in various interactive applications, including educational games.

5.1 Theoretical Background: The Core Methods

In this section, we outline those specific concepts and methods that will serve as a basis for the development of theoretical and practical framework for executing automatically controlled morphing between 2D shapes with textures. We will introduce representational schemes for 2D heterogeneous objects as well as the specific methods called space-time blending (STB) and space-time transfinite interpolation

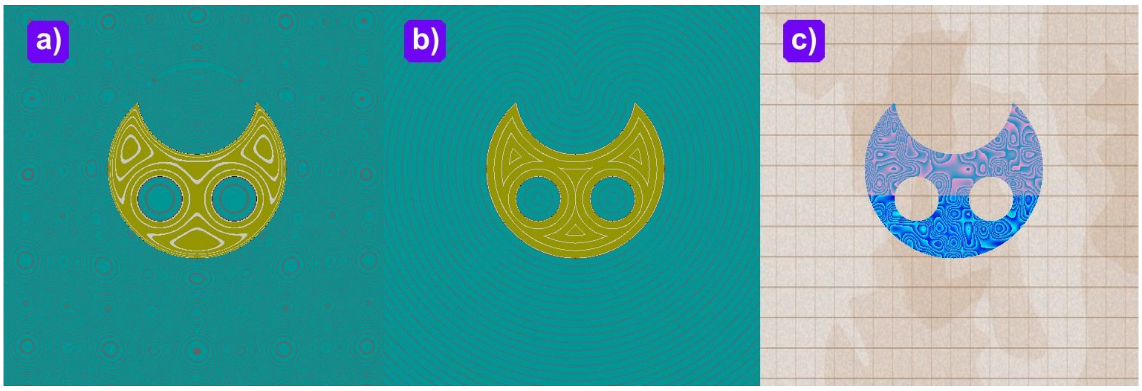


Figure 5.2: *The process of generating HFRep with attributes: a) a FRep defined shape is used as an input; b) an input shape is converted to HFRep; c) for the obtained HFRep a simple segmentation of the geometric shape is done with further attribute definition using procedural noise functions parameterised by the distance.*

(STTI). These methods will serve as the basis for a novel method for solving the automatic metamorphosis problem without establishing any correspondences between two images or functionally defined heterogeneous objects (see section 5.2) as well as for several mathematically substantiated new technical solutions for more visually impressive results (see section 5.3).

5.1.1 Object Representations

In this chapter we mainly consider two representations for defining a geometric shape of the input and target heterogeneous objects. They are SDF and HFRep. We will briefly outline how input images can be converted to SDFs and how we can generate HFRep objects with attributes.

If the input and target shapes are given as 2D raster images as it can be seen in Fig. 5.1 (a), we can convert them to the SDF representation in two steps. First, we need to binarise the input image (see Fig. 5.1, b) to extract the boundary of the shape along with its interior. Then we compute the distance to the extracted shape (see Fig. 5.1, c) using, for example, one of the signed distance transform (SDT) algorithms discussed in subsection 2.3.5.

On the contrary, if we are dealing with HFRep objects, first, we have to define the desired shape using FRep functions. Then, according to the basic algorithm defined for HFRep in section 3.6, we need to compute the unsigned distance field for the generated FRep object, for example shown in Fig. 5.2 (a). This can be done using the distance transform algorithm. Finally, we need to restore the sign for the HFRep field using the FRep field of the initial object. The generated HFRep field is shown in Fig. 5.2 (b).

The attributes in interior of the obtained HFRep geometric shape can be defined using a procedural texture. Fig. 5.2 (c) shows a heterogeneous HFRep object with coloured wooden texture that was obtained using a procedural function $f_{wood}(\boldsymbol{p})$. This function is constructed using hash table htab($\boldsymbol{p}$) allowing for random sampling of the position values $\boldsymbol{p}$ multiplied by the frequency ν. The procedural function for the wood can be defined as follows:

$$g(\boldsymbol{p}) = \mathrm{htab}(\boldsymbol{p} \cdot \nu) \cdot c; \quad f_{wood}(\boldsymbol{p}) = g(\boldsymbol{p}) - int(g(\boldsymbol{p})); \tag{5.1}$$

where $c > 1$ is a constant, $g(\boldsymbol{p})$ is a noise function, $int(g(\boldsymbol{p}))$ is an integer part of

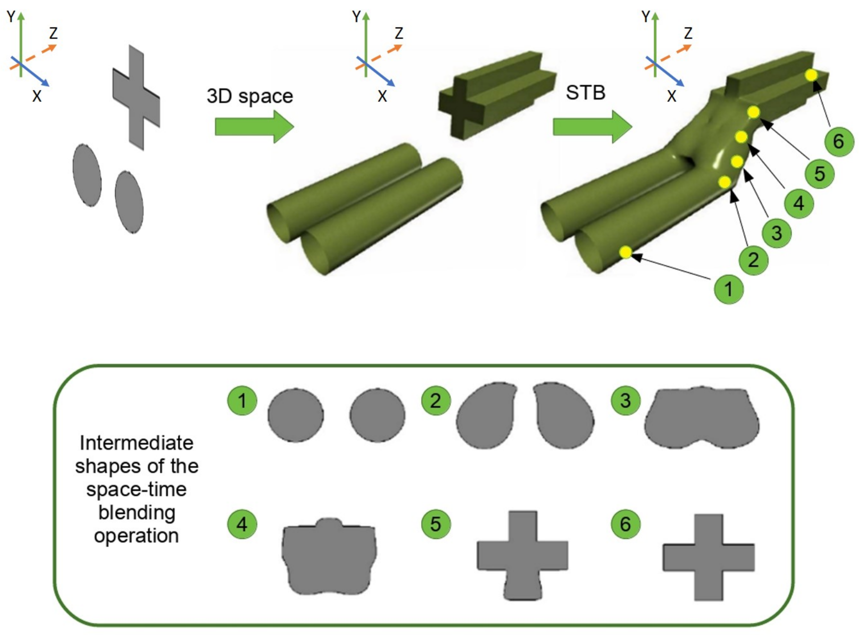


Figure 5.3: *The concept of Space-Time Blending: two given 2D shapes (two disks and a cross, top left) are extended to 3D space as half-cylinders (top centre) with a gap between them. Then a blending union operation is applied, adding material (top right). Intermediate shapes are presented by cross-sections (bottom centre).*

the function $g(\boldsymbol{p})$ output value. To parameterise $f_{wood}(\boldsymbol{p})$ by the distance, we assign the distance values to the frequency parameter ν.

Then a simple segmentation of the geometric shape of the objects was done. We split the shape into four regions and assign colours using the obtained HFRep distance function $F_{HFRep}(\boldsymbol{p})$ and procedural function $f_{wood}(\boldsymbol{p})$ that defines the texture of the wood.

5.1.2 Space-Time Blending

As it was mentioned in the introduction to this chapter, the input and target objects can be defined as a 2D raster images or as a 2D distance-based (SDFs, HFReps, ADFs, IDFs) heterogeneous objects. These objects can be of an arbitrary topology. The main method that we are going to use for the realisation of metamorphosis between such objects without any prior knowledge of their correspondences will be space-time blending (STB) (Pasko, Pasko, and Kunii 2005). In its essence (see Fig. 5.3), STB is a geometric operation of bounded blending performed in a higher dimensional space. If we consider 2D shapes in that higher dimensional space, the Z axis will be associated with time t. Blending between the initial shape S_1 and the target shape S_2 (see Fig. 5.5) happens in the time interval $t \in [0, 1]$, where, while at $t \leq 0$ only the first shape S_1 exists, then at $t > 0$ it disappears and at time $t = 1$ the second shape appears and exists for any $t \geq 1$. Let us outline the basics of this method.

Let us introduce two input shapes S_1 and S_2 represented in that higher dimensional space by distance-based functions $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$ respectively. Then the resulting function $F_b(f_{1_t}, f_{2_t}, f_{3_t})$ representing the blending between those two

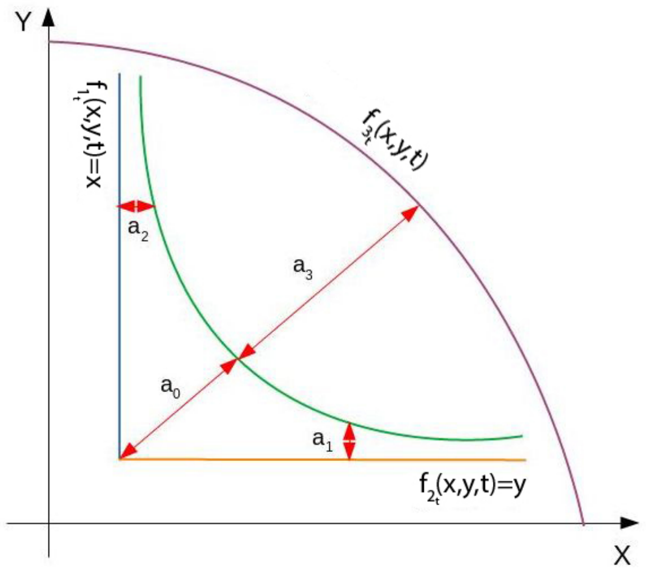


Figure 5.4: *The intersection of two objects defined by the equations $f_{1_t}(x,y,t) = x$ and $f_{2_t}(x,y,t) = y$, restricted by the bounding solid represented by the function $f_{3_t}(x,y,t) = 1 - \left(\frac{x}{4}\right)^2 - \left(\frac{y}{4}\right)^2$ defining the bounding disk. This is a simple illustration of the geometrical meaning of coefficients a_i, $i \in [0,3]$.*

shapes is:

$$F_b(f_{1_t}, f_{2_t}, f_{3_t}) = F(f_{1_t}, f_{2_t}) + a_0 \cdot disp_b(d_r(f_{1_t}, f_{2_t}, f_{3_t})), \tag{5.2}$$
$$F(f_{1_t}, f_{2_t}) = f_{1_t}(x,y,t) + f_{2_t}(x,y,t) + \sqrt{f_{1_t}^2(x,y,t) + f_{2_t}^2(x,y,t)}$$

where $d_r(f_{1_t}, f_{2_t}, f_{3_t})$ is a generalised distance function and $F(f_{1_t}, f_{2_t})$ is a set-theoretical union of two shapes defined by the R-functions introduced by Rvachev (Rvachev 1973). The resulting shape S_2 obtained using the STB function $F_b(f_{1_t}, f_{2_t}, f_{3_t})$ is affected by the bounding solid defined by the function $f_{3_t}(x,y,t)$. The bounding solid is a functionally defined object which restricts the area in which the actual blending happens. In most cases it is convenient to use an intersection of two cutting planes which can be seen in Fig. 5.5. In the case discussed here, the blending operation will be bounded only along the time axis.

The displacement function $disp_b(d_r(f_{1_t}, f_{2_t}, f_{3_t}))$ is set up as

$$disp_b(d_r(f_{1_t}, f_{2_t}, f_{3_t})) = \begin{cases} \frac{(1-d_r^2(f_{1_t},f_{2_t},f_{3_t}))^3}{1+d_r^2(f_{1_t},f_{2_t},f_{3_t})}, & d_r(f_{1_t}, f_{2_t}, f_{3_t}) < 1 \\ 0, & otherwise \end{cases}, \tag{5.3}$$

where $d_r^2(f_{1_t}, f_{2_t}, f_{3_t})$ is defined as

$$d_r^2(f_{1_t}, f_{2_t}, f_{3_t}) = \begin{cases} \frac{d_{r_1}^2}{d_{r_1}^2 + d_{r_2}^2}, & d_{r_2} > 0 \\ 1, & otherwise \end{cases}, \tag{5.4}$$
$$d_{r_1}^2(f_{1_t}, f_{2_t}) = \left(\frac{f_{1_t}(x,y,t)}{a_1}\right)^2 + \left(\frac{f_{2_t}(x,y,t)}{a_2}\right)^2,$$
$$d_{r_2}^2(f_{3_t}) = \begin{cases} \left(\frac{f_{3_t}(x,y,t)}{a_3}\right)^2, & f_{3_t}(x,y,t) > 0 \\ 0, & otherwise \end{cases} \tag{5.5}$$

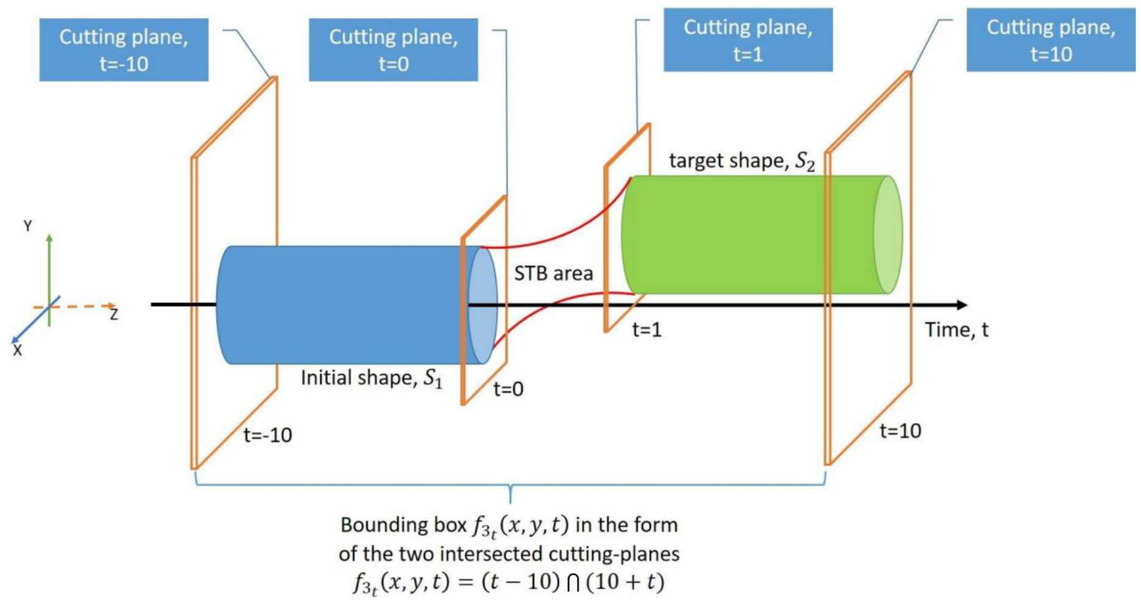


Figure 5.5: *The Space-Time Blending scheme.*

The function $d_{r_1}(f_{1_t}, f_{2_t})$ is a generalised distance function between two shapes $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$. This function $d_{r_1}(f_{1_t}, f_{2_t})$ provides the algebraic distance measure to both initial S_1 and target shapes S_2. The function $d_{r_2}(f_{3_t})$ controls the influence of the function $f_{3_t}(x, y, t)$ on the overall shape of the blend.

Coefficients $a_0, a_1, a_2, a_3 \in \mathbb{R}$ are non-zero numerical parameters. To describe their geometrical meaning let us consider the example shown in Fig. 5.4. Let us assume that $f_{1_t}(x, y, t) = x$, $f_{2_t}(x, y, t) = y$ and as a bounding solid we use disk $f_{3_t}(x, y, t) = 1 - \left(\frac{x}{4}\right)^2 - \left(\frac{y}{4}\right)^2$. We apply the STB method to these. The coefficient a_0 defines the total displacement of the blending surface from two initial surfaces S_1 and S_2, defined by $f_{1_t}(x, y, t)$, $f_{2_t}(x, y, t)$ and controls how much material will be added or subtracted from the overall blend (see Fig. 5.4). Coefficients a_1 and a_2 are proportional to the algebraic distance between the blending surface and the original surfaces S_1 and S_2 defined by $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$, and control the blend symmetry (see Fig. 5.4). The coefficient a_3 is proportional to the algebraic distance between the blending surface and the surface of the bounding solid (see Fig. 5.4) and controls the influence of function $f_{3_t}(x, y, t)$ on the overall blend.

It is not always intuitive to describe a complex shape using FRep. That is why we suggest to use signed distance fields (SDFs) in a discrete form introduced in subsection 2.3.5, to define both shapes S_1 and S_2. This representation allows to describe any type of geometrical shapes of any complexity in a predictable way.

In the 2D case, the basic STB algorithm can be described as follows:

Step 1. Two shapes S_1 and S_2 are defined by the functions $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$. These shapes are represented by scalar fields on the X, Y plane (see Fig. 5.3, top left). Note that both shapes can have radically different topology.

Step 2. Each shape is used as a cross section of a half-cylinder in 3D space, by extending them in the Z dimension, with a gap in-between (see Fig. 5.3, top middle).

Step 3. The blending union operation adding material to the gap is applied to the two half-cylinders (see Fig. 5.3, top right). The blending operation can be user controlled by varying a_i coefficients.

Step 4. The Z dimension is used as time t. The cross sections of the blending process along the Z axis between the source S_1 and target S_2 shapes are marked with numbers (see Fig. 5.3, top right) and are shown at the bottom of Fig. 5.3.

The implementation in 3D is essentially the same, but raises the shapes into 4D and uses the fourth axis to represent time t.

5.1.3 Space-Time Transfinite Interpolation

As we are not only dealing with geometry but also with textures, we need a specific technique for texture transformations. This means that in addition to a geometric shape metamorphosis, it is essential to independently define colour and other attribute transformations taking into account the initial and in-between shapes. One of the possible ways was described in (Sanchez, Fryazinov, Adzhiev, et al. 2015), where the authors proposed to apply transfinite interpolation (Rvachev et al. 2001) to attributes such as colour. The proposed method was called space-time transfinite interpolation (STTI) as it deals with functionally defined objects with specified attribute distributions. The main idea of STTI is that two scalar fields defining an initial and a target shapes are used for calculating normalised weights $\omega_1(x, y, t)$ and $\omega_2(x, y, t)$ in equation (5.6) that are further used for computing intermediate colours.

Let us introduce mathematical expressions for the STTI which we are going to use in the next sections. Let us assume that the initial shape is S_1 and that the target shape is S_2, which are defined by the functions $f_{1_t}(x, y, t)$ and $f_{2_t}(x, y, t)$. The attribute interpolation between single-partitioned objects can then be defined as follows:

$$\boldsymbol{c}(x, y, t) = \omega_1(x, y, t)\boldsymbol{c}_1 + \omega_2(x, y, t)\boldsymbol{c}_2; \tag{5.6}$$
$$\omega_1(x, y, t) = \frac{f_{2_t}(x, y, t)}{f_{1_t}(x, y, t) + f_{2_t}(x, y, t)}; \quad \omega_2(x, y, t) = \frac{f_{1_t}(x, y, t)}{f_{1_t}(x, y, t) + f_{2_t}(x, y, t)}$$

where $\boldsymbol{c}_1$ is the colour of the input shape S_1, $\boldsymbol{c}_2$ is the colour of the target shape S_2, $t \in [0, 1]$ and $\omega_1(x, y, t)$ as well as $\omega_2(x, y, t)$ are weights. The following constrains should be satisfied:

- at the initial time step, $t = 0$: $\omega_1(x, y, 0) = 1; \omega_2(x, y, 0) = 0; f_{1_t}(x, y, 0) \geq 0;$
- at the final time step, $t = 1$: $\omega_1(x, y, 1) = 0; \omega_2(x, y, 1) = 1; f_{2_t}(x, y, 1) \geq 0;$
- $\omega_1(x, y, t) + \omega_2(x, y, t) = 1.$

In the general case, if the given shapes S_1 and S_2 have complex textures with multiple semi-disjoint partitions with a constant attribute $\tilde{\boldsymbol{c}}_j$ assigned to j^{th} partition, the contribution of these partitions to the resulting attribute $c_i(x, y)$ at a given external point (x, y) is:

$$c_i(x, y) = \frac{\sum_{j=1}^{N} \tilde{w}_j(x, y)\tilde{\boldsymbol{c}}_j}{\sum_{j=1}^{N} \tilde{w}_j(x, y)}; \quad \tilde{w}_j(x, y) = \frac{1}{\widehat{f}_j(x, y)}, \quad i = 1, 2 \tag{5.7}$$

where N is the number of partitions, $\tilde{w_j}(x,y)$ is the weight of each partition and $\widehat{f_j}(x,y)$ is a function defining the quadratic Euclidean distance from the external point (x,y) to the pixel in interior of the object. In the current work we consider textured objects as single-partitioned objects and use the equation (5.7) for computing the interpolated colours $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, where N in this case is the number of pixels inside the textured shape.

5.2 Automatically Controlled 2D Heterogeneous STB

In this section we describe in a step-by-step manner a novel method for solving the problem of metamorphosis between 2D heterogeneous objects for generating in-between textured shapes without the need to set any correspondences. The suggested method combines the techniques described in section 5.1. However, as our practical example demonstrates, a basic algorithm for using those techniques in their standard form does not produce completely satisfactory results. We will, therefore, identify the drawbacks of these techniques.

5.2.1 Heterogeneous STB: Statement of The Problem

Let us formulate the problem of metamorphosis between two objects with textures as follows: given two textured objects with topologically arbitrary 2D shapes defined by SDFs or HFReps $f_1(x,y)$ and $f_2(x,y)$, we aim at realising an automatic metamorphosis between those objects with a smooth transition between both geometric shapes as well as their textures without establishing any correspondences between them.

5.2.2 Description of The Basic Metamorphosis Method

Let us provide a systematic description of the method. Initial and target shapes S_1 and S_2 are represented as 2D images on a monocolored background, e.g. black or white or as a HFRep scalar fields with defined attributes in interior of the shape and some constant colour assigned in exterior of the shape. The steps of the algorithm are as follows:

Step 1. If at this step two input shapes S_1 and S_2 are given as images, then we convert both of them to binary images to extract the shapes. Then we apply one of the signed distance transform methods that we have reviewed in subsection 2.3.5 to obtain discrete SDFs $F_{SDF_1}(\boldsymbol{p}) : \boldsymbol{p} \mapsto F_{SDF_1}(\boldsymbol{p}), \forall \boldsymbol{p} \in \Omega$ and $F_{SDF_2}(\boldsymbol{p}) : \boldsymbol{p} \mapsto F_{SDF_2}(\boldsymbol{p}), \forall \boldsymbol{p} \in \Omega$ representing those images in a pixel domain Ω where $\boldsymbol{p} = (x,y)$ is a pixel. The sign of both functions $f_1(x,y)$ and $f_2(x,y)$ changes on the boundary of the binarised shapes.

If two input shapes S_1 and S_2 are defined using an HFRep heterogeneous object representation $O_{H_{V,HFRep}}$, equation (3.19), then we have already obtained smooth signed distance fields for these shapes. In this case, we can skip the next step of the basic algorithm and proceed to the third step. At this step we also have to convert HFReps with defined

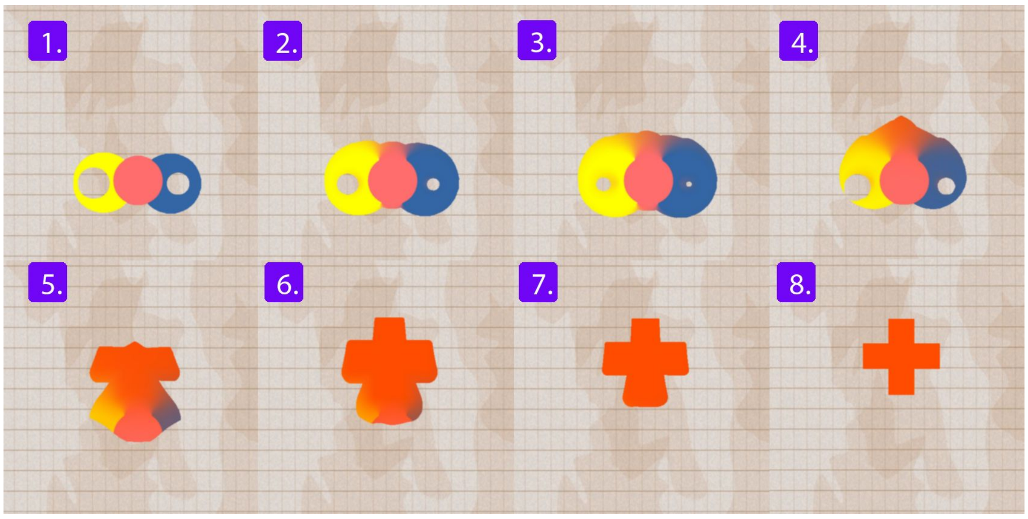


Figure 5.6: *The result of applying the basic algorithm to compute the metamorphosis between two shapes. Supplementary video: figure5.6.mpg*

attributes into 2D images that will be further processed at the last step of the basic algorithm.

Step 2. For smoothing the discrete SDFs $F_{SDF_1}(x, y)$, $F_{SDF_2}(x, y)$ and converting them into a continuous representation in the form of the functions $f_1(x, y)$ and $f_2(x, y)$, we apply an interpolation procedure between the discrete SDFs values.

Step 3. The next step is applying STB to shapes S_1 and S_2 represented by the SDFs or HFReps functions $f_1(x, y)$ and $f_2(x, y)$ in a continuous form for generating intermediate frames of the metamorphosis. The texture transformation is not considered yet as STB works only with geometry.

Step 4. To obtain the texture transformations we apply the STTI method described in subsection 5.1.3 to the input images. We calculate the sum of the colour attributes weighted by the distance from the current pixel to other pixels using equation (5.7) without establishing correspondences. The resulting images can be created with a transparent background and superimposed on to any background image.

5.2.3 Advantages and Drawbacks of The Basic Method

Fig. 5.6 demonstrates how the suggested method works. It shows a sequence of frames, demonstrating metamorphosis between an initial shape S_1 represented as three overlapping disks with different topology and various colours, and a target shape S_2 which is a red cross. We use the equations (5.2)-(5.4) with manually set coefficients a_0, a_1, a_2 and a_3. As there was no automatic procedure for setting those coefficients we had to experimentally select the values $a_0 = 2, a_1 = 1.3, a_2 = 1.5$ and $a_3 = 1$, which resulted in the most acceptable results. The outcome of this example allows us to make the following conclusions regarding advantages and drawbacks of the suggested method. The method has the following obvious advantages:

- It allows for generating in-between shapes in terms of both geometry and textures;

- Initial and target shapes S_1 and S_2 can be defined using images or distance-based representational schemes, e.g. SDFs or HFReps, that are capable to define heterogeneous objects.
- Initial and target shapes S_1 and S_2 can be of an arbitrary complexity in terms of geometry and topology. In particular, they do not need to have similar shapes and can include disjoint components;
- Initial and target shapes S_1 and S_2 can overlap each other in-place or they can be separated in space. The latter is a major issue when attempting cross-dissolving.

The following drawbacks pertain to this method:

- A problematic smooth transition: some in-between shapes (see Fig. 5.6, frames 1-3) appear overly similar, whilst there is an obvious non-smooth jump-wise transition between some neighbouring shapes (see Fig. 5.6, frames 4-5). This is true for both geometry and colour;
- An additional unexpected inflation in some in-between shapes appears (see Fig. 5.6, frames 3, 5);
- The manual control for choosing the working set of parameters is problematic, especially for a non-specialist users. To realise an adequate automatic STB control one needs to have a set of well-defined default parameters a_0, a_1, a_2 and a_3 in equations (5.2)-(5.4). However, the method for specifying those parameters, especially taking into account specific features of the initial and target objects, has not been developed yet.

In the next section a number of substantiated solutions for the mentioned problems will be described.

5.3 New Techniques Enhancing The Basic Method

In this section we are going to introduce new techniques for solving the drawbacks of the method for automatic metamorphosis between 2D textured shapes introduced in section 5.2. A solution for these can be achieved by either using one of the following techniques or a combination of them. Let us briefly outline them:

- Half-cylinders smoothing: in subsection 5.1.2 it was mentioned that both initial and target shapes are considered as cross-sections of half-cylinders in 3D space. As the STB method assumes that half-cylinders are defined using set-theoretical operations, a rapid change in the gradient of the resulting shape in the time interval $t \in [0, 1]$ can appear because of the influence of those operations. This rapid change visually results in a 'jump' during shape metamorphosis, which can be seen in Fig. 5.6, frames 4-5. This problem can be solved by smoothing sharp-edges of half-cylinders by additionally applying STB to them using equations (5.2)-(5.4).
- Automatic control of the STB parameters: it is non-trivial for a non-expert user to select the satisfactory values for coefficients a_0, a_1, a_2 and a_3 defined in equations (5.2)-(5.4). To simplify this we introduce an algorithm for the automatic control of the coefficients a_0, a_1, a_2 and a_3. For this we use both

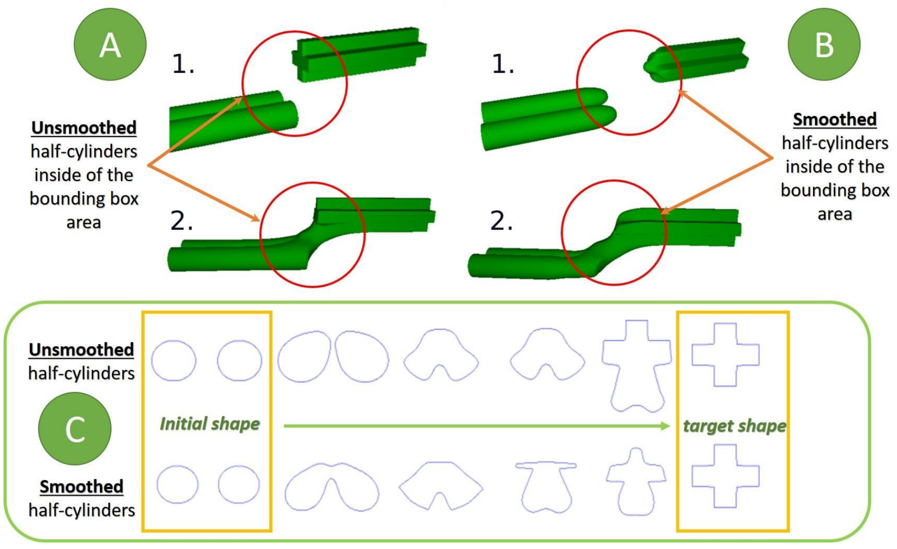

A
1.
2.
Unsmoothed half-cylinders inside of the bounding box area
B
1.
2.
Smoothed half-cylinders inside of the bounding box area
Unsmoothed half-cylinders
C
Smoothed half-cylinders
Initial shape
target shape

Figure 5.7: *The blending operation conducted between two circles and a cross that are extended to 3D space. A) The result of the blending operation before applying the smoothing operation; B) The result of the blending operation after applying the smoothing operation; C) in-between cross-sections for the unsmoothed (C, top) and smoothed cases (C, bottom).*

image processing based techniques and interval arithmetic based estimations for the coefficient a_0 as well as exploiting the geometric meaning of the generalised distance $d_{r_1}(f_{1_t}, f_{2_t})$ in equation (5.4).

- New bounding solid functions: some undesirable additional inflation of the shape during metamorphosis can appear even with automatic control of the coefficients a_0, a_1, a_2 and a_3. For better shape transformation control between the initial and target objects, we suggest to use two specific functions, each defining a new bounding solid. One of these functions defines a truncated cone and the other defines a truncated pyramid.

- Affine translation: we need to introduce a certain restriction on the distance between initial and target shapes S_1 and S_2. If the distance exceeds the defined limit, STB can produce disjoint in-betweenings. The problem can be solved by applying an affine translation to either initial shape S_1 or to target shape S_2 to satisfy the defined distance limit between them. In addition to this, an affine translation can be used as a shape inflation control while conducting the metamorphosis using the suggested method described in subsection 5.2.2.

All four proposed techniques will be discussed in details in this section.

5.3.1 Smoothing of The Half-Cylinders

One of the drawbacks of STB is the presence of fast transitions or 'jumps' in the $t \in [0, 1]$ interval (see Fig. 5.5) in which the most significant shape transformation happens. The cause of this is that when STB is applied to a half-cylinder, the result is bounded by a plane orthogonal to the time axis. The set-theoretic subtraction of the half-space from the infinite cylinder results in a sharp edge for a half-cylinder boundary (see Fig. 5.7, A) and the sharp edge remains a significant feature in the resulting blend (see Fig. 5.7, A (bottom)). To avoid this effect, instead of using the set-theoretic subtraction of the half-space, we will use a bounded blending subtraction. Smoothing can be achieved by applying a blending intersection operation, removing material between the infinite cylinder and the bounding planar half-space for both shapes (see Fig. 5.7, B):

$$F_b(f_{1_t}, f_{2_t}, f_{3_t}) = F(f_{1_t}, f_{2_t}) + a_0 \cdot disp_b(d_r(f_{1_t}, f_{2_t}, f_{3_t})); \tag{5.8}$$
$$F(f_{1_t}, f_{2_t}) = f_{1_t}(x,y,t) + f_{2_t}(x,y,t) - \sqrt{f_{1_t}(x,y,t)^2 + f_2(x,y,t)^2};$$

where $F_b(f_{1_t}, f_{2_t}, f_{3_t})$ is the resulting blending function defining a smoothed half-cylinder, $disp_b(d_r(f_{1_t}, f_{2_t}, f_{3_t}))$ is the displacement defined in equations (5.3)-(5.4), and $F(f_{1_t}, f_{2_t})$ is a FRep set-theoretical intersection operation. If we compare the two bottom pictures (A), (2) and (B), (2) in Fig. 5.7, we notice that smoothing the sharp edges of the cylinders results in a smooth transition between two shapes (B, bottom).

The smoothing process for initial and target shapes S_1 and S_2 can be described as follows:

$$S_1: \; f_{1_t}(x,y,t) = f_{DF_1}(x,y,t); \quad f_{2_t}(x,y,t) = -t; \tag{5.9}$$
$$f_{3_t}(x,y,t) = t + P_c, \quad t \geq -P_c;$$
$$S_2: \; f_{1_t}(x,y,t) = f_{DF_2}(x,y,t); \quad f_{2_t}(x,y,t) = t - 1; \tag{5.10}$$
$$f_{3_t}(x,y,t) = P_c - t, \quad t \leq P_c;$$

where $f_{1_t}(x,y,t)$ is the SDF or HFRep function raised into three-dimensional space defining either the initial shape S_1 or the target shape S_2, $f_{2_t}(x,y,t)$ is the function defining a smoothing object as the subtraction of the negative half-space along time-axis t, $f_{3_t}(x,y,t)$ is the bounding function restricting the area where the STB bounding intersection happens and P_c is the position of the cutting plane on the time-axis t. These functions are substituted in equation (5.8) for obtaining the smoothed result. For smoothing both the initial shape S_1 as well as the target shape S_2, we need to manually set the coefficients a_0, a_1, a_2 and a_3.

In lower half of Fig. 5.7 (part C), we compare the results before (C, top) and after (C, bottom) application of the suggested smoothing method to both the initial shape S_1 (two circles) and the target shape S_2 (a cross). As it follows from the figure, after applying the smoothing operation to the half-cylinders, 'jumps' between the frames have disappeared and there are no more similar in-between shapes.

5.3.2 Automatic Control of Space-Time Blending Parameters

Automatic control of the coefficients a_0, a_1, a_2 and a_3 allows us to make our approach more user-friendly. Here we describe the derivation of the expressions for estimating the coefficients a_0, a_1 and a_2. We intend to make these coefficients geometry-dependent for better control of the blending shape in case of both in-place as well as spaced image morphing. These coefficients should be greater or equal to 1 to guarantee that a transition between the initial and target shapes is visible. This procedure depends on the geometrical properties of both initial and target objects and the geometrical meaning of the coefficients a_0, a_1 and a_2 described in subsection 5.1.2.

Let us assume that $a_3 = 1$. Then, according to equation (5.4) it does not affect the bounding solid defined by the function $f_{3_t}(x,y,t)$. At the initial step we need

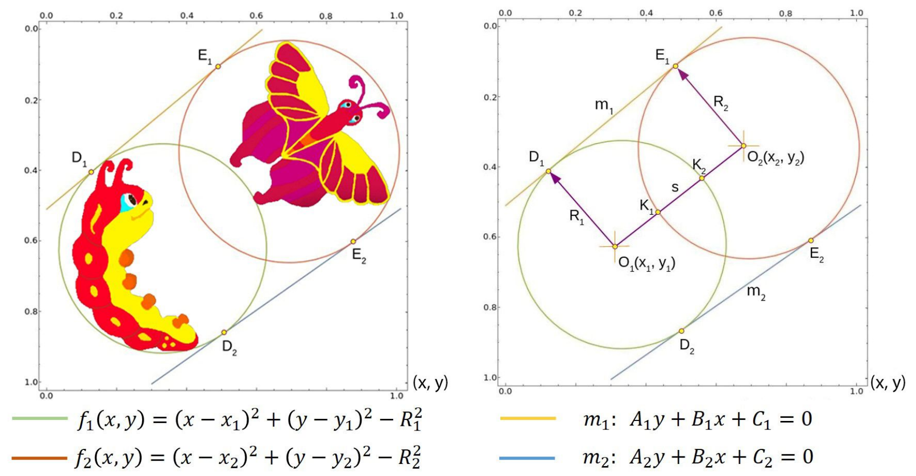

E1
D1
E2
D2
(x, y)
m1
R2
R1
K1
K2
s
O2(x2, y2)
O1(x1, y1)
m2
$f_1(x, y) = (x - x_1)^2 + (y - y_1)^2 - R_1^2$
$f_2(x, y) = (x - x_2)^2 + (y - y_2)^2 - R_2^2$
m_1: $A_1y + B_1x + C_1 = 0$
m_2: $A_2y + B_2x + C_2 = 0$

Figure 5.8: *Geometrical scheme for estimating the coefficients a_0 and $a_1 = a_2$, where S_1 is the initial shape and S_2 is the target shape.*

to find the circumscribed circles around initial shape S_1 and target shape S_2 shown in Fig. 5.8. It is not essential that these two circles overlap. The radii of the circumscribed circles R_1 and R_2 and their centre coordinates O_1 and O_2 are known. As the input data is represented by the images, all the calculations are made in a normalised coordinate system: $x = x/I_w$; $y = y/I_h$, where I_w and I_h are image width and height.

The proposed estimation instead of using the bounded blending function $F_b(f_{1_t}, f_{2_t}, f_{3_t})$ defined in equation (5.2), relies on a blending equation (Pasko, Pasko, and Kunii 2005) without time dependence introduced in the following form:

$$F_{blend}(f_1, f_2) = f_1(x,y) + f_2(x,y) + \sqrt{f_1(x,y)^2 + f_2(x,y)^2} + \frac{a_0}{1 + d_{r_1}(f_1, f_2)} \quad (5.11)$$

$$d_{r_1}(f_1, f_2) = \sqrt{\left(\frac{f_1(x,y)}{a_{1,2}}\right)^2 + \left(\frac{f_2(x,y)}{a_{1,2}}\right)^2}$$

where $f_1(x,y)$ and $f_2(x,y)$ are functions with distance property and $d_{r_1}(f_1, f_2)$ is a generalised distance between the initial and target shapes represented by functions $f_1(x,y)$ and $f_2(x,y)$.

The estimation process starts from finding the values for coefficients a_1 and a_2. For simplification let us assume that the blending process will be symmetric. This means that $a_1 = a_2 = a_{1,2}$. As stated in (Pasko, Pasko, and Kunii 2005), coefficients $a_1 > 0$ and $a_2 > 0$ are proportional to the algebraic distance between the blending surface and the original surfaces S_1 and S_2 defined by functions $f_1(x,y)$ and $f_2(x,y)$.

To estimate coefficient $a_{1,2}$ we suggest the following algorithm: the generalised distance $d^2_{r_1}(f_1, f_2)$ used in equation (5.11) is equal to the quadratic distance $||s||^2$ (see Fig. 5.8) between two shapes S_1 and S_2. To calculate this distance we need to solve two equation systems for both circles as well as segment O_1O_2 to define the coordinates of points K_1 and K_2. Then the distance $||s||^2$ between K_1 and K_2 can be defined as:

$$||s||^2 = (x_{K_2} - x_{K_1})^2 + (y_{K_2} - y_{K_1})^2 \quad (5.12)$$

Let us assume that that the generalised distance $d^2_{r_1}(f_1(x_{K_2}, y_{K_2}), f_2(x_{K_1}, y_{K_1}))$ between shape S_1 and shape S_2 is equal to the quadratic Euclidean distance $||s||^2$

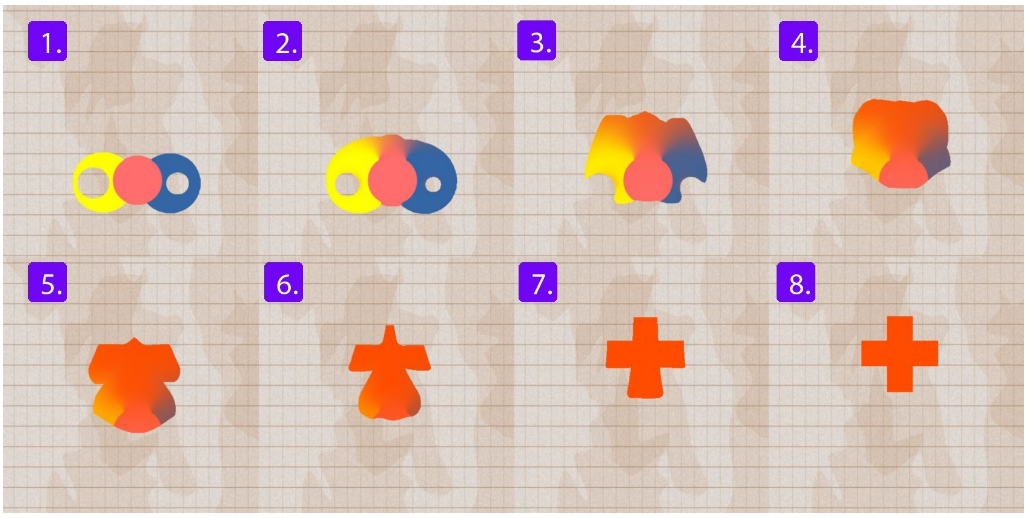


Figure 5.9: *Blending between shape S_1 consisting of three circles, two of which have a hole, and an orange cross S_2 using the STB and STTI techniques. Supplementary video: figure5.9.mpg*

between these two shapes (see Fig. 5.8). Then we obtain the final estimation for $a_1 = a_2 = a_{1,2}$:

$$a_{1,2} = \sqrt{\frac{f_1(x_{K_2}, y_{K_2})^2 + f_2(x_{K_1}, y_{K_1})^2}{||s||^2}} \tag{5.13}$$

To estimate coefficient a_0 we suggest to use interval arithmetic (Moore, Kearfott, and Cloud 2009). Here we provide an outline of the solution for the interval of the coefficient a_0. Let us assume for simplicity that two input shapes S_1 and S_2, defined by functions f_1 and f_2 respectively, are represented by circumscribed circles (To simplify further we will use f_1 and f_2). We bound the blending operation defined by equation (5.11) using two tangential lines m_1 and m_2 (see Fig. 5.8), which can be written in the general case as:

$$m_1(x, y) = A_1x + B_1y + C_1; \quad m_2(x, y) = A_2x + B_2y + C_2$$

where A_1, B_1, C_1 and A_2, B_2, C_2 are constants that can be obtained by solving the following system of equations:

$$\begin{cases} A_ix_1 + B_iy_1 + C_i = R_1 \\ A_ix_2 + B_iy_2 + C_i = R_2 \\ A_i^2 + B_i^2 = 1 \end{cases}$$

where $i = 1, 2$, (x_1, y_1) are the coordinates of point O_1, (x_2, y_2) are the coordinates of point O_2, R_1 is the radius of the circumscribed circle with the centre point O_1 and R_2 is the radius of the circumscribed circle with the centre point O_2.

After solving this system the following coefficients are obtained:

$$\begin{aligned} A_{2,1} &= R_mX_m \pm Y_m\sqrt{1 - R_m^2}; \\ B_{2,1} &= R_mY_m \mp X_m\sqrt{1 - R_m^2}; \\ C_{2,1} &= R_1 - (A_{2,1}x + B_{2,1}y), \end{aligned}$$

and the following coefficients X_m, Y_m and R_m are defined as:

$$X_m = \frac{x_2 - x_1}{D_m}; \quad Y_m = \frac{y_2 - y_1}{D_m}; \quad R_m = \frac{R_2 - R_1}{D_m}; \quad D_m = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

The indices for $A_{2,1}, B_{2,1}$ and $C_{2,1}$ are obtained using system of equations (5.14) and are inverted here because we use the normalised coordinate system in which all the calculations are conducted.

The final intervals for a_0 can be written in a simplified form assuming that the intervals for the functions $[f_1] = [f_{1_{min}}, f_{1_{max}}]$ and $[f_2] = [f_{2_{min}}, f_{2_{max}}]$ are as follows:

$$\begin{cases} a_0 \leq (A_1x + B_1y + C_1 - (f_1 + f_2 + \sqrt{f_1^2 + f_2^2}))(1 + \frac{1}{a_{1,2}^2}(f_1^2 + f_2^2)), \\ a_0 \geq (A_2x + B_2y + C_2 - (f_1 + f_2 + \sqrt{f_1^2 + f_2^2}))(1 + \frac{1}{a_{1,2}^2}(f_1^2 + f_2^2)) \end{cases} \tag{5.14}$$

For calculating the final interval this inequality equations system (5.14) should be converted to the interval representation. First let us define the minimum and maximum values for x and y taking into account that all the calculations are represented in a normalised coordinate system.

$$\begin{aligned} x_{min} &= \min(x_1, x_2) - R_i,\ \ x_{max} = \max(x_1, x_2) + R_i, \\ y_{min} &= \min(y_1, y_2) - R_i,\ \ y_{max} = \max(y_1, y_2) + R_i, \end{aligned} \tag{5.15}$$

The index i for the radius R_i is chosen through the index of the variable obtained using the min / max operation. The intervals for the following functions for the two circles, written in the normalised form

$$f_i = \sqrt{x^2 - 2x_ix + x_i^2} + \sqrt{y^2 - 2yy_i + y_i^2} - R_i; \quad i = 1, 2$$

can be written as

$$\begin{aligned} [f_1] = \Bigg[& \sqrt{x_{min}x_{max} - 2x_1x_{max} + x_1^2} + \sqrt{y_{min}y_{max} - 2y_1y_{max} + y_1^2} - R_1, \\ & \sqrt{x_{max}^2 - 2x_1x_{min} + x_1^2} + \sqrt{y_{max}^2 - 2y_1y_{min} + y_1^2} - R_1 \Bigg], \\ [f_2] = \Bigg[& \sqrt{x_{min}x_{max} - 2x_2x_{max} + x_2^2} + \sqrt{y_{min}y_{max} - 2y_2y_{max} + y_2^2} - R_2, \\ & \sqrt{x_{max}^2 - 2x_2x_{min} + x_2^2} + \sqrt{y_{max}^2 - 2y_2y_{min} + y_2^2} - R_2 \Bigg], \end{aligned} \tag{5.16}$$

The inequality system of equations (5.14) can be rewritten using interval arithmetic as

$$\begin{aligned} a_0 \leq [& \min[g_{1_{min}}b_{1_{max}}, g_{1_{max}}b_{1_{min}}, g_{1_{min}}b_{1_{min}}, g_{1_{max}}b_{1_{max}}], \\ & \max[g_{1_{min}}b_{1_{max}}, g_{1_{max}}b_{1_{min}}, g_{1_{min}}b_{1_{min}}, g_{1_{max}}b_{1_{max}}]] \\ a_0 \geq [& \min[g_{2_{min}}b_{1_{max}}, g_{2_{max}}b_{1_{min}}, g_{2_{min}}b_{1_{min}}, g_{2_{max}}b_{1_{max}}], \\ & \max[g_{2_{min}}b_{1_{max}}, g_{2_{max}}b_{1_{min}}, g_{2_{min}}b_{1_{min}}, g_{2_{max}}b_{1_{max}}]] \\ [g_1] &= [A_1x_{min} + B_1y_{min} + C_1 - f_{s_{max}}, A_1x_{max} + B_1y_{max} + C_1 - f_{s_{min}}] \\ [g_2] &= [A_2x_{min} + B_2y_{min} + C_2 - f_{s_{max}}, A_2x_{max} + B_2y_{max} + C_2 - f_{s_{min}}] \\ [b_1] &= \left[\left(f_{1_{min}}f_{1_{max}} + f_{2_{min}}f_{2_{max}} \right) \frac{1}{a_{1,2}^2} + 1, \left(f_{1_{max}}^2 + f_{2_{max}}^2 \right) \frac{1}{a_{1,2}^2} + 1 \right] \\ [f_{s_{min}}, f_{s_{max}}] &= [f_{1_{min}} + f_{2_{min}} + \sqrt{f_{1_{min}}f_{1_{max}} + f_{2_{min}}f_{2_{max}}}, \\ & \quad f_{2_{max}} + f_{2_{max}} + \sqrt{f_{1_{max}}^2 + f_{2_{max}}^2}] \end{aligned} \tag{5.17}$$

The final interval for a_0 can then be obtained depending on the following cases:

- if $a_0 \leq [a_{0_{min_1}}, a_{0_{max_1}}] \cap a_0 \geq [a_{0_{min_2}}, a_{0_{max_2}}]$ and $a_{0_{min_1}} < 0; a_{0_{min_2}} < 0$ then the final interval will be $a_0 \in [a_{0_{max_1}}, a_{0_{max_2}}]$;
- if $a_0 \leq [a_{0_{min_1}}, a_{0_{max_1}}] \cap a_0 \geq [a_{0_{min_2}}, a_{0_{max_2}}]$ and $0 \leq a_{0_{min_1}} \leq 1; 0 \leq a_{0_{min_2}} \leq 1$ then the final interval will be $a_0 \in [1, \max(a_{0_{max_1}}, a_{0_{max_2}})]$;
- if $a_0 \leq [a_{0_{min_1}}, a_{0_{max_1}}] \cap a_0 \geq [a_{0_{min_2}}, a_{0_{max_2}}] = \varnothing$ this may be an overestimation for a_0 and the final interval will be formed as $a_0 \in [a_{0_{min_1}}, a_{0_{max_2}}]$; $a_{0_{min_1}} \geq 1$.

Fig. 5.9 shows an example of applying the smoothing operation to the half-cylinders with automatic control of the coefficients a_0, a_1, a_2 and a_3. Comparing these results with those shown in Fig. 5.6, generated using the basic algorithm described in subsection 5.2.2, one can conclude that here the transition between initial shape S_1 and target shape S_2 is smooth and there is no additional inflation within in-betweens.

In case if circumscribed circles are not overlapping and their centres coincide, the estimation procedure for a_0 defined by system of inequations (5.14) should be slightly modified. The coefficients $a_1 = a_2 = a_{1,2}$ should be set to one and inequality (5.14) will be rewritten as:

$$a_0 \leq ((x - x_i)^2 + (y - y_i)^2 - R_i^2 - (f_1 + f_2 + \sqrt{f_1^2 + f_2^2}))\Big(1 + \frac{1}{a_{1,2}^2}\Big(f_1^2 + f_2^2\Big)\Big),$$

where $i = 1, 2$ and depending on the chosen radius of the biggest circumscribed circle $\max(R_1, R_2)$. In this case the blending area is restricted by the circumscribed circle with the biggest radius. Then using interval arithmetic we can obtain the final interval:

$$\begin{aligned} a_0 \leq [\,&\min[f_{c_{min}} b_{1_{max}}, f_{c_{max}} b_{1_{min}}, f_{c_{min}} b_{1_{min}}, f_{c_{max}} b_{1_{max}}], \\ &\max[f_{c_{min}} b_{1_{max}}, f_{c_{max}} b_{1_{min}}, f_{c_{min}} b_{1_{min}}, f_{c_{max}} b_{1_{max}}]] \\ [f_c] = [&(x_{min} - x_i)(x_{max} - x_i) + (y_{min} - y_i)(y_{max} - y_i) - \max(R_1^2, R_2^2), \\ &(x_{max} - x_i)^2 + (y_{max} - y_i)^2 - \max(R_1^2, R_2^2)] \\ i = &\begin{cases} 1, & if \;\; \max(R_1^2, R_2^2) = R_1^2 \\ 2, & if \;\; \max(R_1^2, R_2^2) = R_2^2 \end{cases}. \end{aligned} \tag{5.18}$$

where interval for $[b_1]$ was introduced in equation (5.17).

5.3.3 Truncated Cone and Truncated Pyramid as a Bounding Solid

An additional inflation caused by the lack of the STB control can be reduced using a different technique. We suggest using a tighter bounding solid $f_{3_t}(x, y, t)$, represented as either a truncated cone or as a truncated pyramid, equations (5.10).

We assume that the circumscribed circles have been already obtained for both initial object S_1 and target object S_2. To avoid shape inflation, we specify that the entire metamorphosis process happens within the new bounding solid defined by function $f_{3_t}(x, y, t)$.

Let us introduce a truncated cone with circumscribed circles around initial and target shapes S_1 and S_2 as its two bases. To obtain the defining function $f_{3_t}(x, y, t)_{cone}$ for the truncated cone (Fig. 5.10, a), a simple linear interpolation

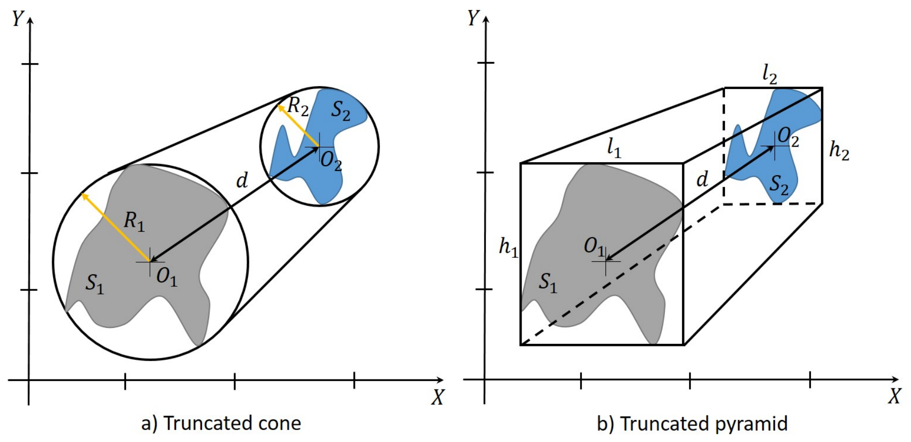


Figure 5.10: *New functions for defining the bounding solid $f_{3_t}(x, y, t)$: a) Truncated cone; b) Truncated pyramid. S_1 is the initial object and S_2 is the target object.*

between the centres and radii of the circles is applied, where the parameter k is defined in the interval $k \in [0, 1]$, which is dependent on time t:

$$f_{3_t}(x, y, t)_{cone} = R_m^2 - (x - \widehat{X}_m)^2 - (y - \widehat{Y}_m)^2 \tag{5.19}$$
$$R_m = R_1 + (R_2 - R_1)k, \quad \widehat{X}_m = x_{c_1} + (x_{c_2} - x_{c_1})k,$$
$$\widehat{Y}_m = y_{c_1} + (y_{c_2} - y_{c_1})k, \quad k = \frac{t - t_1}{t_2 - t_1}, \quad t_1 = -10; \quad t_2 = 10.$$

Here (x_{c_1}, y_{c_1}) is the centre coordinates O_1 of the circumscribed circle around initial shape S_1, (x_{c_2}, y_{c_2}) is the centre coordinates O_2 of the circumscribed circle around target shape S_2, R_1 and R_2 are the radii of the two circumscribed circles and t_1 and t_2 are the minimum and the maximum values of the interval time t. We use the values recommended for t_1 and t_2 from the article on STB (Pasko, Pasko, and Kunii 2005).

A truncated pyramid is defined (Fig. 5.10, b) with circumscribed rectangular bounding boxes for initial and target shapes S_1 and S_2 as its two bases. For this case we need to define each base of the pyramid as a result of the FRep set-theoretic intersection operation and linear interpolation between widths the l_1, l_2, the heights h_1, h_2 and the centres of the rectangles.

$$f_{3_t}(x, y, t)_{pyramid} = (L_m - |x - \widehat{X}_m|) \cap (H_m - |y - \widehat{Y}_m|) \tag{5.20}$$
$$L_m = l_1 + (l_2 - l_1)k, \quad H_m = h_1 + (h_2 - h_1)k,$$
$$\widehat{X}_m = x_{r_1} + (x_{r_2} - x_{r_1})k, \quad \widehat{Y}_m = y_{r_1} + (y_{r_2} - y_{r_1})k,$$
$$k = \frac{t - t_1}{t_2 - t_1}, \quad t_1 = -10; \quad t_2 = 10.$$

Here (x_{r_1}, y_{r_1}) and (x_{r_2}, y_{r_2}) are the coordinates of the rectangles' centres O_1 and O_2.

To define a new function for the bounding solid as either a truncated cone or a truncated pyramid we need to use the FRep set-theoretic intersection operation $\cap$ between either the cone defined by $F_{solid}(x, y, t) = f_{3_t}(x, y, t)_{cone}$ or the pyramid defined by $F_{solid}(x, y, t) = f_{3_t}(x, y, t)_{pyramid}$ with the two time-cutting planes $(t + 10)$ and $(10 - t)$

$$f_{3_t}(x, y, t) = F_{solid}(x, y, t) \cap (t + 10) \cap (10 - t) \tag{5.21}$$

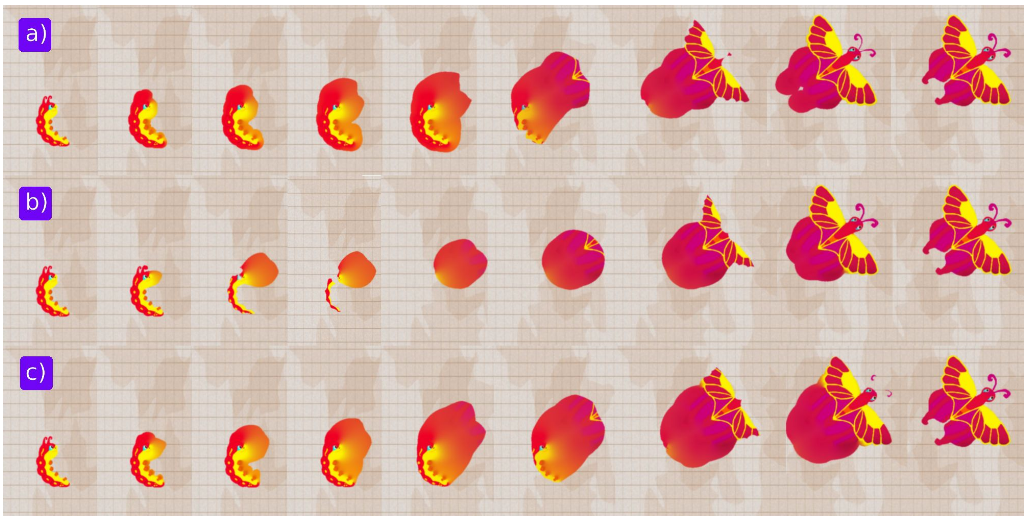


Figure 5.11: *Blending operation conducted between two shapes S_1 (caterpillar) and S_2 (butterfly), using a) two half-planes as bounding solid function $f_{3_t}(x,y,t)_{planes}$; b) truncated pyramid as bounding solid function $f_{3_t}(x,y,t)_{pyramid}$; c) truncated cone as bounding solid function $f_{3_t}(x,y,t)_{cone}$. Supplementary videos: figure5.11(a).mpg, figure5.11(b).mpg, figure5.11(c).mpg*

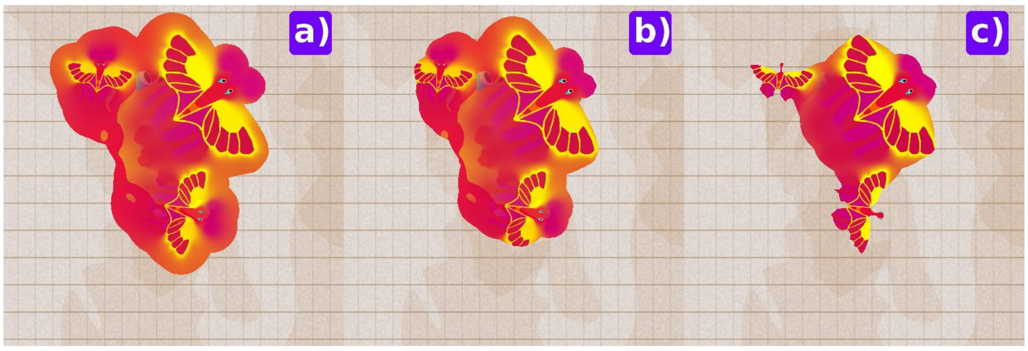


Figure 5.12: *Blending operation conducted between two shapes S_1 (caterpillar), S_2 (three butterflies) in case of in-place morphing using a) two half-planes as bounding solid function $f_{3_t}(x,y,t)_{planes}$; b) truncated pyramid as bounding solid function $f_{3_t}(x,y,t)_{pyramid}$; c) truncated cone as bounding solid function $f_{3_t}(x,y,t)_{cone}$. Supplementary videos: figure5.12(a).mpg, figure5.12(b).mpg, figure5.12(c).mpg*

In Fig. 5.11 we present the results of applying three different methods with automatic control of the coefficients a_0, a_1 and a_2: (a) is the result of using two half-planes defined by function $f_{3_t}(x,y,t)_{planes} = (t+10) \cap (10-t)$, (b) is the result of using the truncated pyramid defined by function $f_{3_t}(x,y,t)_{pyramid}$ and (c) is the result of using the truncated cone defined by function $f_{3_t}(x,y,t)_{cone}$. The metamorphosis shape obtained using the two half-planes defined by function $f_{3_t}(x,y,t)_{planes}$ is quite similar to the metamorphosis shape obtained using function $f_{3_t}(x,y,t)_{pyramid}$, set up as a truncated pyramid.

From Fig. 5.11 follows:

- Using the truncated pyramid (see Fig. 5.11, c) as the new bounding box provides a more visually accurate result compared to that one obtained to the two half-plane bounding solid (see Fig. 5.11, a), but it produces a slightly more inflated result.
- Using the truncated cone (see Fig. 5.11, b) as the new bounding box results

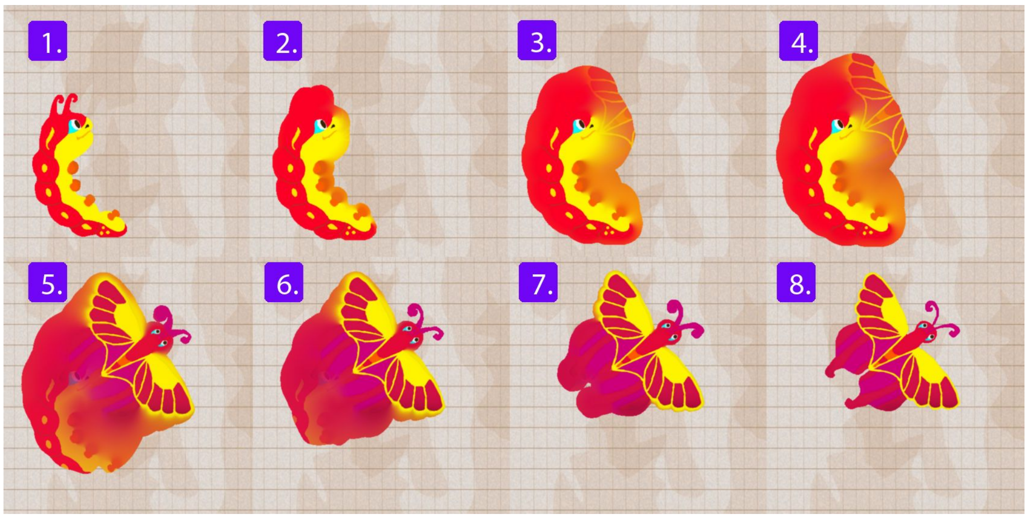


Figure 5.13: *Blending operation conducted between two shapes S_1 (caterpillar) and S_2 (butterfly) using affine translation. Supplementary video: figure5.13.mpg*

in less inflation. The in-between shapes in this case are also quite different. Note that for the truncated cone, better (smoother) results can be obtained when the number of frames in the animation sequence is increased.

The criterion for using one of the suggested new bounding boxes is as follows: if the distance d (see Fig. 5.10) between the centres of two circumscribed circles satisfies the condition $d \leq (R_1 + R_2)$, the blending shape obtained with automatic estimation of coefficients a_i, $i = 0, 1, 2, 3$ might still have some additional inflation. In case of an in-place animation, for example, when there is one big object transformed into several smaller ones (see Fig. 5.12, a), it is recommended to apply one of the introduced bounding solids. As it follows from Fig. 5.12 (b), using the truncated pyramid produces slightly less inflated results compared to the bounding solid defined as two bounding planes in Fig. 5.12 (a). Using the truncated cone provides even better results (see Fig. 5.12, c).

5.3.4 Affine Translation for Space-time Blending Shape Control

If two shapes S_1 and S_2 are placed too far away from each other, there is a chance that disconnections in in-between shapes might appear. To avoid this, shape S_1 can be left static and target shape S_2 should be shifted closer to shape S_1 or vice versa. This can be achieved by applying an affine translation to shape S_2.

The criterion for applying the affine translation is the following: if the distance d (see Fig. 5.10) between the centres of two circumscribed circles satisfies the condition $d \gg (R_1 + R_2)$, where R_1 is the radius of the first circle and R_2 is the radius of the second circle, then the two objects S_1 and S_2 are too far from each other. In this case there is a chance that the algorithm for automatic estimation of the coefficients a_i, $i = 0, 1, 2, 3$ will not produce satisfying results, so additional enhancement is essential.

Affine translation allows for reducing the additional shape inflation during the STB metamorphosis. According to the algorithm presented in subsection 5.2.2, at the first step we need to calculate SDT for both input and target shapes S_1 and S_2. Then the geometric centres of the two shapes are superimposed at the initial time

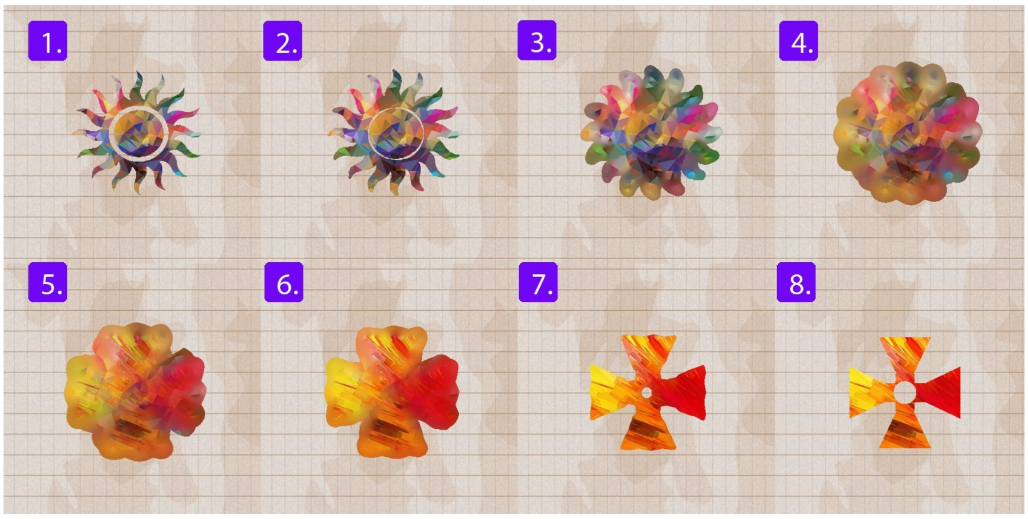


Figure 5.14: *Metamorphosis between two textured shapes using SDFs, STB and STTI, where initial shape S_1 is a textured sun and target shape S_2 is a textured cross. Supplementary video: figure5.14.mpg*

step. If two shapes are inside of each other, we need to calculate the interval for a_0 according to equation (5.18). Then, we need to check whether the shifted shape S_2 is still within the borders of the circumscribed circle around the static shape S_1. This can be achieved by solving the system of equations for both circles.

Two tangential lines can be drawn to these two circles when this system have two non-coincident real solutions. In that case the algorithm for automatic control of coefficient a_0 described in subsection 5.3.2 can be used for the rest of the motion of shape S_2. After defining a_0, a_1 and a_2, at each step, we apply mapping of the SDT values from the initial position of shape S_2 to its current position to compute the in-between shapes.

After each SDT mapping is done, STB is calculated according to equations (5.2)-(5.4) and an affine translation is applied to the result. The image sequence obtained after applying the affine translation can be seen in Fig. 5.13, where the butterfly (target shape, S_2) is emerging from the caterpillar (static initial shape S_1) while moving towards its initial position. This operation does not produce any additional shape inflation. At each step we use automatic control for recalculation of the coefficients a_0, a_1 and a_2 described in subsection 5.3.2 and apply a smoothing half-cylinder operation to both shapes as described in subsection 5.3.1.

5.4 Algorithmic Implementation of Automatically Controlled Heterogeneous STB

In this section we describe the implementation of the basic algorithm introduced in subsection 5.2.2 with enhancing techniques that we have described in section 5.3. We have implemented our algorithm using *C++*, the *OpenCV* (Intel 2020) library for the basic image handling and *OpenMP* (OpenMP Architecture Review Board 2020) for multi-threading. All examples were computed on a laptop with a 2.6 GHz *Intel Skylake 6700* processor and 16 Gb of RAM. The implementation of the method is available on the GitHub .

https://github.com/teshaTe/2D-metamorphosis

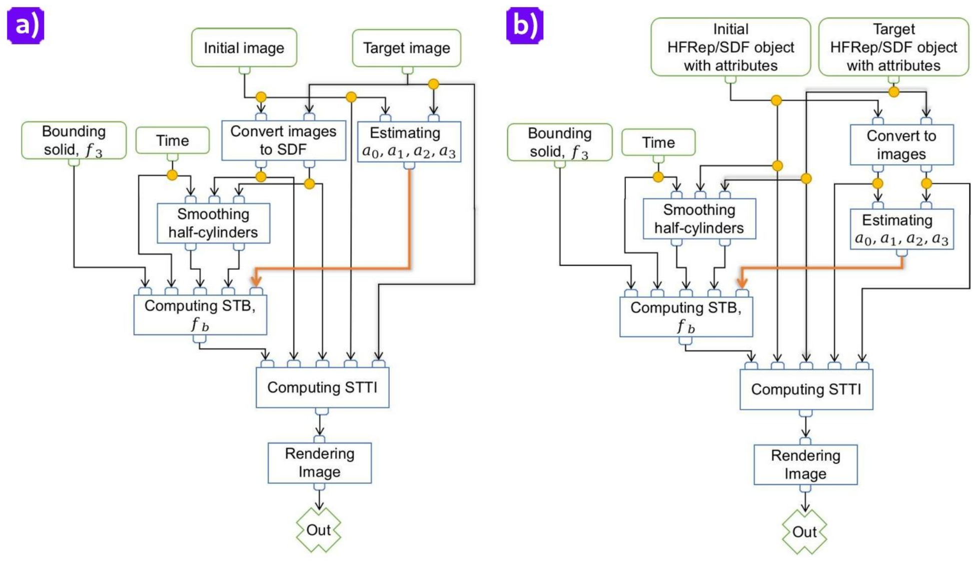


Figure 5.15: *The algorithmic scheme of the 2D heterogeneous STB. a) The algorithm for the case when input and target object are defined by the images; b) The algorithm for the case when input and target objects are heterogeneous objects defined by the HFRep or SDF functions;*

Let us outline the enhanced basic algorithm with technical details and recommendations at which step to apply the introduced in this chapter techniques following the Fig. 5.15 with algorithm diagrams (a) and (b). The algorithm (a) corresponds to the case when the input and target shapes are defined as raster images, while the algorithm (b) corresponds to the case when input and target shapes are represented as an HFRep or SDF scalar fields with specified attribute distributions.

Step 1. First we define the input and target geometric shapes S_1 and S_2 with textures. They can be presented as images (see algorithm in Fig. 5.15, a) or using HFReps or SDFs with specified attribute distributions (see algorithm in Fig. 5.15, b).

Step 2. Then, if the input and target geometric shapes S_1 and S_2 with textures are given as images (see Fig. 5.15, a), first, we have to binarise them to extract the geometric shapes. Then we can apply the signed sequential Euclidean distance transform (SSEDT) which implementation was discussed in subsection 4.1.1 to compute SDFs for each image. The computed SDFs are discrete. We suggest to smooth them using some spline interpolation (e.g. B-splines or bicubic splines) to convert them to a continuous distance field.

Otherwise, if the input and target geometric shapes S_1 and S_2 are defined using HFReps or SDFs with specified attribute distributions (see Fig. 5.15, b), first, we need to compute images with textures for them that will be used during the next steps of the algorithm. If the geometric shapes are defined using HFReps, we do not need to additionally smooth them as this operation was already applied during the construction procedure of HFRep. In case of SDFs, we suggest to apply same smoothing operation as we have discussed earlier at this step.

Step 3. At this step we will use images of the input and target geometric shapes

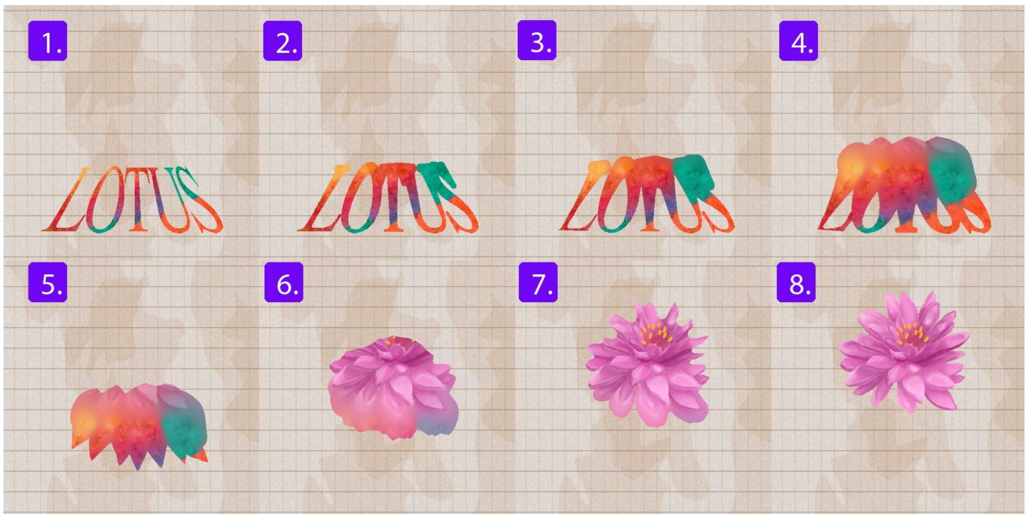


Figure 5.16: *Metamorphosis between two textured shapes S_1 (text 'LOTUS') and S_2 (lotus flower) using SDFs, STB and STTI. Supplementary video: figure5.16.mpg*

S_1 and S_2 for estimating STB parameters a_0, a_1, a_2 and a_3. According to the algorithm described in subsection 5.3.2, first, we have to find circumscribed circles around the geometric shapes S_1 and S_2. This is done using OpenCV library. Then we follow the mathematical outline described in subsection 5.3.2 to compute the parameters. In general case, we assume, that parameter $a_3 = 1$. Note that the textures are not considered at this stage.

Step 4. At this step we apply the smoothing half-cylinders algorithm described in subsection 5.3.1 to both input and target shapes S_1 and S_2 represented with SDFs or HFReps.

Step 5. Then we need to define one of the bounding solids defined by the function $f_3(x_1, x_2, t)$ (t is the time) that will be further used in the STB computation at the next step. Here we can apply one of the following bounding solids: two half-planes, truncated pyramid or truncated cone. To construct truncated pyramid we have to find circumscribed circles around input and target geometric shapes S_1 and S_2. To construct truncated cone we have to find circumscribed rectangles around input and target geometric shapes S_1 and S_2. The rest of computations are made according to subsection 5.3.3. Truncated pyramid and truncated cone could significantly reduce the undesired additional inflation of the inbetween shapes.

Step 6. The next step is applying the space-time blending (STB) to shapes S_1 and S_2 represented by functions $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ that were smoothed using smoothing half-cylinders technique applied at the fourth step. This step is applied according to the equations (5.4)-(5.3). The texture transformation is not considered yet as STB works only with geometry.

Step 7. To get intermediate shapes with textures we apply the STTI method. We calculate the sum of the colour attributes weighted by the distance between a currently sampled pixel and the rest of the image pixels using equation (5.7) without establishing any correspondences between

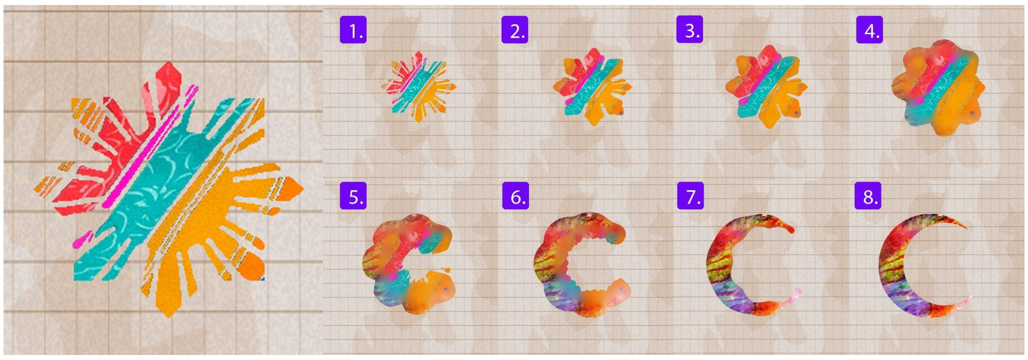


Figure 5.17: *Metamorphosis between a multiply subdivided, textured star shape S_1 and a textured moon S_2 using the SDFs, STB and STTI techniques for a radical topology change. Supplementary video: figure5.17.mpg*

coloured partitions. The resulting images can be made with transparent background and superimposed on any background image.

Step 8. Finally, we can render the result as 2D images.

On the basis of some experiments that we conducted, we suggest setting up the parameters for automatic control in case of applying one of the bounding solids (truncated pyramid or truncated cone) as follows:

- for the truncated cone coefficient $a_3 = 0.03 \cdot \min(R_1, R_2)$; for smoothing of the half-cylinders the coefficients should be: for the initial shape S_1: $a_0 = -0.8$, $a_1 = a_2 = a_3 = 1$ and for the target shape S_2: $a_0 = -0.5$, $a_1 = a_2 = a_3 = 1$.

- for the truncated pyramid coefficient $a_3 = 0.13 \cdot \min(R_1, R_2)$; for smoothing of the half-cylinders the coefficients should be: for the initial shape S_1: $a_0 = -0.3$, $a_1 = a_2 = a_3 = 1$ and for the target shape S_2: $a_0 = -0.5$, $a_1 = a_2 = a_3 = 1$.

If it is essential to use affine translations described in subsection 5.3.4, it should be applied between the third and fourth step of the suggested algorithm. Note, that coefficients a_0, a_1, a_2 and a_3 should be evaluated at each step of shape translation using the algorithm described in subsection 5.3.2.

5.5 Applications and Results

In this subsection we will discuss the results of application of the proposed method, shown in Figs. 5.9, 5.11, 5.13, 5.14, 5.16, 5.19, 5.21 and 5.22. In table 5.1 we present the comparative timing results in frames per second (FPS) to demonstrate that interactive frame rates have been achieved.

For the examples shown in Figs. 5.9, 5.14, 5.16, 5.17, 5.19, 5.21 and 5.22 we use the following values as coefficients a_0, a_1, a_2 and a_3 for smoothing half-cylinders (see subsection 5.3.1), which were obtained from a number of conducted experiments. On the basis of this we suggest the following values for the initial shape S_1: the position for the cutting plane $P_c = 5$ and coefficients $a_0 = -0.3$, $a_1 = a_2 = a_3 = 1$. For the target shape S_2 we suggest the following values: the position for the cutting plane $P_c = 5$ and coefficients $a_0 = -0.5$, $a_1 = a_2 = a_3 = 1$.

Execution: Sequential / Parallel / Parallel + Scaled Image					
Figure	Average FPS	Min FPS	Max FPS	a_0 interval	$a_{1,2}$
Fig. 5.9	1.69 / 3.37 / 18.82	40.19/ 32.33 / 46.25	0.10 / 0.22 / 2.82	[2.49, 3.85]	1.12
Fig. 5.11					
a)	0.78 / 1.64 / 13.46	38.98 / 31.08 / 42.38	0.08 / 0.17 / 2.21	[0.48, 3.26]	1.78
b)	1.23 / 2.42 / 16.18	38.32 / 30.61 / 42.63	0.14 / 0.29 / 3.76	[0.48, 3.26]	1.78
c)	0.52 / 1.20 / 10.47	35.32 / 27.66 / 40.33	0.07 / 0.17 / 2.22	[0.48, 3.26]	1.78
Fig. 5.13	0.10/ 0.14 / 2.52	0.09/ 0.10 / 1.63	0.02/ 0.03 /1.07	-	-
Fig. 5.14	0.71 / 1.88 / 9.36	33.82 / 27.12 / 41.19	0.05 / 0.13 / 0.34	[-20.22, 2.58]	1.00
Fig. 5.16	0.59 / 1.29 / 12.30	33.70 / 26.88 / 40.20	0.04 / 0.10 / 1.21	[1.75, 2.85]	1.30
Fig. 5.17	0.49 / 1.26 / 12.72	37.36 / 26.40 / 40.32	0.04 / 0.09 / 1.33	[-21.01, 2.57]	1.00
Fig. 5.19	0.88 / 1.65 / 11.04	33.84 / 26.02 / 40.07	0.08 / 0.13 / 1.68	[-9.39, 1.11]	1.76
Fig. 5.21	1.23 / 2.44 / 20.10	34.28 / 29.62 / 41.10	0.06 / 0.14 / 1.93	[-18.58, 2.06]	1.00
Fig. 5.22	0.49 / 1.11 / 12.72	37.36 / 28.61 / 40.32	0.04 / 0.09 / 1.33	[3.57, 3.76]	1.00

Table 5.1: *Comparison table of execution speeds for the image metamorphosis used in the article with timings given as Frame per Second (FPS). In case of an affine translation (see Fig. 5.13) the final two columns are left empty as intervals a_0 and $a_{1,2}$ are dynamically recalculated at each step.*

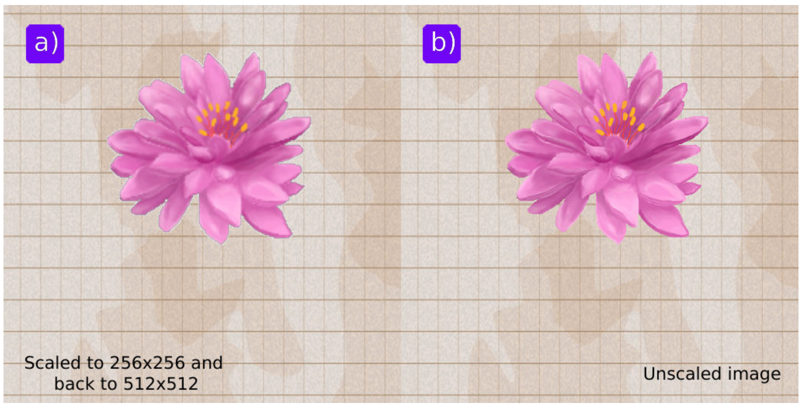


Figure 5.18: *Comparison of the loss in quality between scaled and unscaled images one while conducting metamorphosis; a) image scaled from 512×512 to 256×256 and after application of STB and STTI scaled back to 512×512; b) original picture without scaling.*

In Figs. 5.9, 5.14, 5.16, 5.17, 5.19, 5.21, 5.22 and 5.23 we show the results obtained using the combination of the suggested basic method with automatic control of the coefficients a_0, a_1 and a_2 and smoothing of the half-cylinders. As it follows from the figures, this combination of the techniques produces good quality in-between shapes with a smooth transition between them and without an additional undesired inflation. From table 5.1, it can be seen that this combination of methods works with interactive rates.

In Fig. 5.11 we demonstrate the result obtained using the combination of the suggested basic method with automatic control of the coefficients a_0, a_1 and a_2, smoothing of the half-cylinders and applying two new bounding solids. As discussed in subsection 5.3.3 the result will be slightly inflated when using a truncated pyramid, and the shape of the in-betweens will be different, however, they will be without an additional inflation when using a truncated cone instead. From table 5.1 it can be seen that both new bounding solids slow down the calculation, albeit not drastically.

In Fig. 5.13 we demonstrate the result obtained using the combination of the

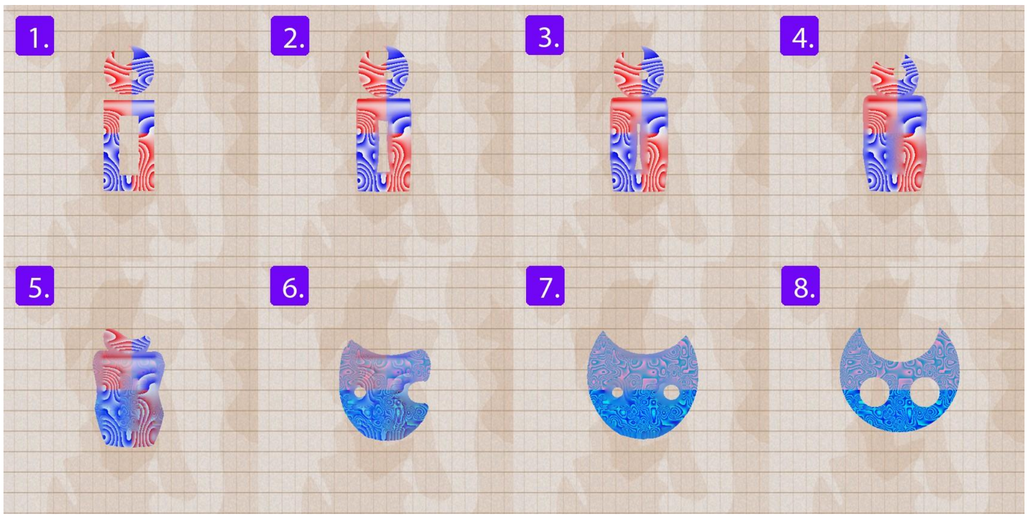


Figure 5.19: *Metamorphosis between two geometric shapes defined by HFRep functions with defined attributes using procedural function of the wood. Supplementary video: figure5.19.mpg*

suggested basic method with automatic control for the coefficients a_0, a_1 and a_2, smoothing the half-cylinders technique and applying an affine translation to the target shape. As it follows from the figure, applying an affine translation produces the results without any additional inflation. Note that according to table 5.1, applying an affine translation can significantly reduce the calculation speed for the whole metamorphosis.

In Figs. 5.16, 5.17, 5.19, 5.21, 5.22 and 5.23 we demonstrate that the introduced method with automatically controlled metamorphosis can handle transformations between multiple objects and objects with complex topologies and different textures while simultaneously preserving smoothness of the transition and without undesired inflation. In Figs. 5.17, 5.19, 5.21, 5.22 and 5.23 we also demonstrate that our method can handle in-place morphing between initial and target shapes, in most cases producing semantically meaningful in-betweens.

In Fig. 5.23 we present the morphing sequence of the textured curved symbol into the textured cross. The texture applied to the curved symbol can be considered as a texture with big changes in colour contrast. As it follows from this figure, our method produces reasonably good results.

In Figs. 5.17 and 5.22 the initial shapes consist of disjoint, coloured patterns that are then morphed into a target shape. In these examples using tiled shapes we imitate the partitioning of textured objects and show that our method produces satisfactory colour interpolation results without establishing any correspondences between partitions. To make the segmentation of the textured shape a fully automatic method, establishment of correspondences between the partitions is required, which is a matter of future work. In table 5.1 we show that the generation of the frame sequences for these examples is relatively fast, even for full image resolution.

In Fig. 5.19 we show the result of the application of the introduced metamorphosis method to two textured HFRep objects defined by the functions. The objects are subdivided into several areas for further definition of the attributes in their interior. The textures in interior of the objects are defined using procedural function of the wood (5.1).

With our method we can achieve interactive frame rates by conducting calculations on scaled images and then scaling them back to full size. This results in a loss

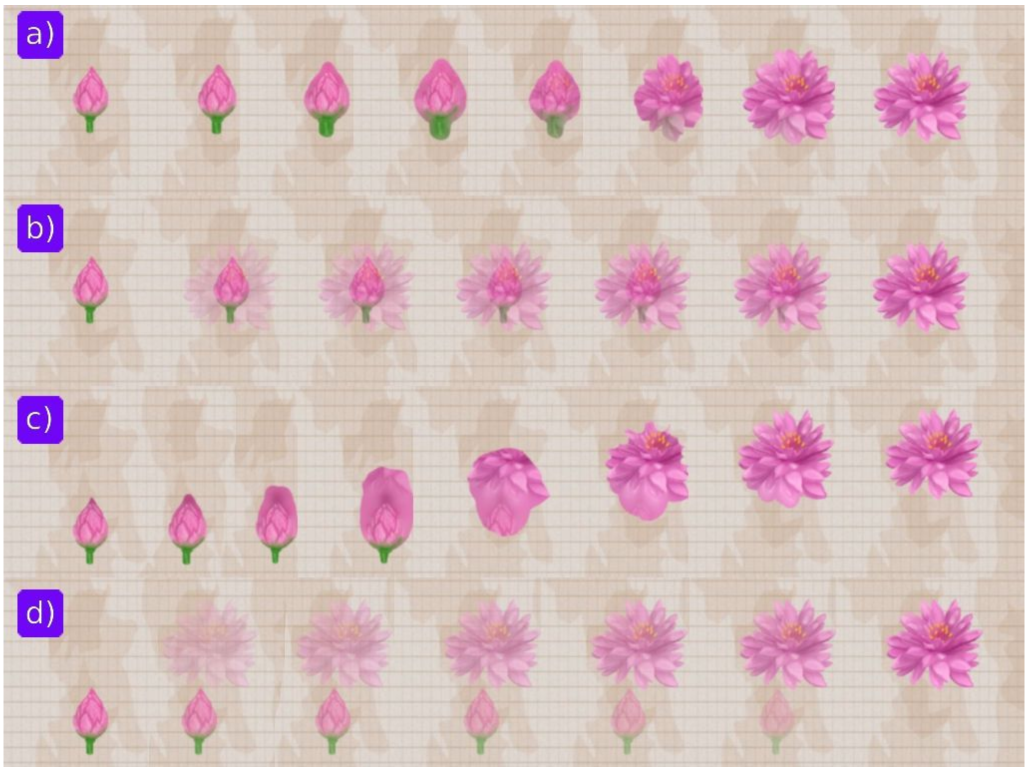


Figure 5.20: *Metamorphosis between two textured shapes using suggested method a), c) and cross-dissolving b), d), where the initial shape S_1 is the bud and the target shape S_2 is the flower. Supplementary videos: figure5.20(a).mpg, figure5.20(b).mpg, figure5.20(c).mpg, figure5.20(d).mpg*

of image quality of around $15 - 20$ percent which can be seen in Fig. 5.18.

As a potential application, we consider an educational game for children with cognitive deficits, including those who are severely disabled. The concept is illustrated in Fig. 5.16. We have already started working on such a much needed game with psychologists.

In Fig. 5.20 we compare the results of using our basic method with automatic control of the coefficients a_0, a_1 and a_2 and smoothing of the half-cylinders with results obtained using the cross-dissolving method. From Fig. 5.20 follows that even though cross-dissolving is a quick linear interpolation, it does not work for gradually changing shapes nor shapes that are placed distantly from each other. Methods working with user defined correspondences can produce more intuitive image transformations, but using these methods is beyond the scope of this work. In addition, providing sensible correspondences in the case of radically different object topology is not an easy or obvious task, especially for non-expert users.

5.6 Conclusions

In this chapter we have presented a novel theoretical and practical method for dealing with 2D textured shapes transformations during metamorphosis. It allows to obtain a smooth transition between source and target textured objects with different topologies without establishing any correspondences.

Most other methods do not provide automatic metamorphosis without establishing some form of correspondences between two given objects or their features. The most relevant methods to compare with in the context of automatically controlled

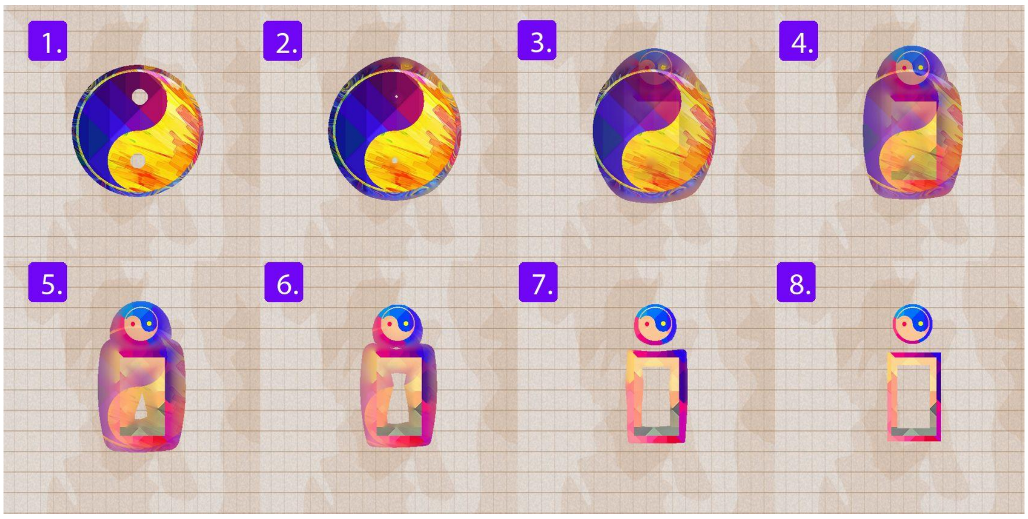


Figure 5.21: *Metamorphosis between two textured shapes using SDFs, STB, STTI techniques and smoothed cylinders, where the round object is the initial shape S_1 and 'i' is the target shape S_2. Supplementary video: figure5.21.mpg*

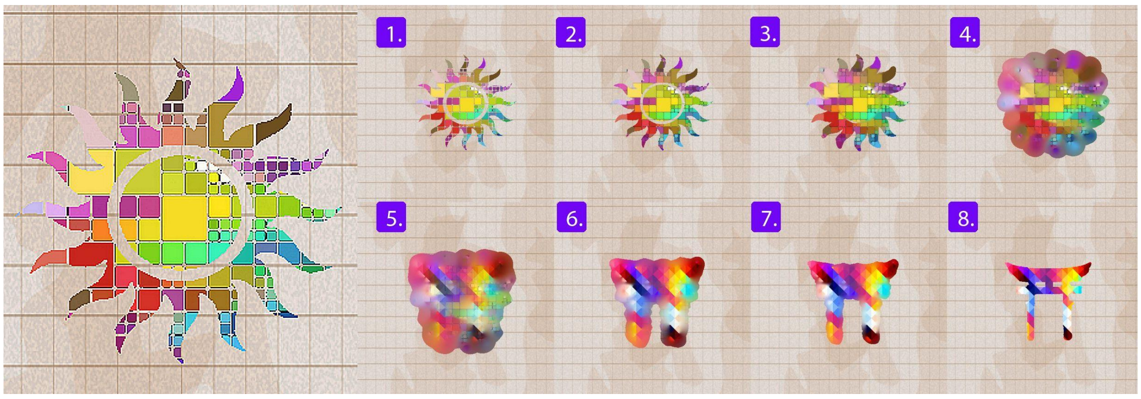


Figure 5.22: *Metamorphosis between two textured shapes S_1 (tiled textured sun) and S_2 (textured ark) using SDFs, STB and STTI techniques for the case of the in-place morphing. Supplementary video: figure5.22.mpg*

metamorphosis are the optimal mass transport based methods, which do handle the morphing in terms of both geometry and images even if the published works do not provide many examples of metamorphosis between textured objects. Our method can produce the in-between frames of reasonable (not always the highest) quality. It is computationally light and oriented towards non-professional users in the context of mainly artistic applications, especially requiring interactive rates, such as games and live streaming.

The proposed basic algorithm relies on a combination of three techniques that were used in the context of the 3D modelling of heterogeneous objects. These techniques are SDFs, STB and STTI. The SDF representation is used for defining functionally based objects. We have shown that HFRep can be also used as one of its hybridisations based on FRep and SDF. In this case, 2D textured shapes are defined using an extended hypervolume representation for HFReps, introduced in section 3.4. The STB technique is used for generating a smooth sequence of the in-betweens and controls only the geometry transformation, while STTI method produces metamorphosed textures within generated intermediate geometric shapes.

We have identified a number of STB drawbacks in our basic algorithm that

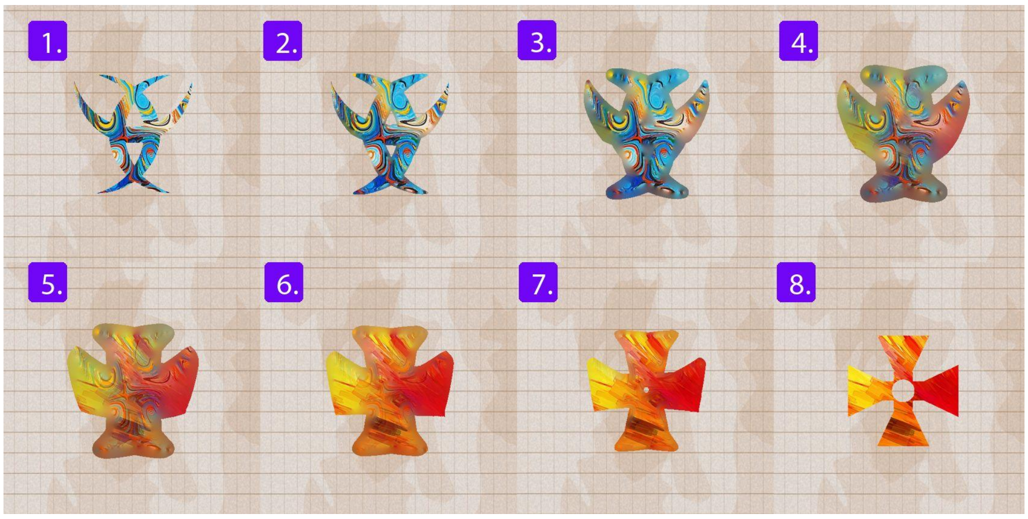


Figure 5.23: *Metamorphosis between two textured shapes S_1 (curved shape with high frequency texture) and S_2 (textured cross) using SDFs, STB and STTI techniques for the case of the in-place morphing. Supplementary video: figure5.23.mpg*

we have addressed by introducing several mathematically substantiated techniques. The half cylinders STB smoothing technique has solved the problem of fast transitions between the initial and target shapes. The STB method is controlled by four parameters a_0, a_1, a_2 and a_3 that are not exactly intuitive to choose for the non-experienced user. Therefore, we have introduced an automatic control for choosing these parameters that allows a user to obtain a visually appealing in-between shapes. For better control of the shape of the in-betweens generated using the STB method, we have introduced two bounding solids, namely a truncated cone and a truncated pyramid. The better shape control of the in-betweens can also be achieved by applying affine transformations to the target shape. These new techniques can be applied separately and in combination.

We have also conducted numerous experiments to find the most suitable values for parameters on which various morphing effects depend. Finally, we have implemented a number of representative test examples proving that the approach does work for in-place morphing, spaced images morphing and that it provides interactive frame rates which is important for many emerging applications. The paper describing this method was published in SIAM Journal on Imaging Sciences (Tereshin, Adzhiev, Fryazinov, Marrington-Reeve, et al. 2020).

Chapter 6

3D Heterogeneous Space-Time Blending in Artistic Applications

In this chapter we present a novel method called the heterogeneous space-time blending (HSTB). This method is an extension of a well-established space-time blending (STB) (Pasko, Pasko, and Kunii 2005) described in detail in the previous chapter. The STB-based technique had been developed to execute 2D metamorphosis (morphing) between two geometric shapes. As to metamorphosis between the attributes of the initial heterogeneous object and the target heterogeneous object, it was implemented using the space-time transfinite interpolation (STTI) (Fryazinov, Sanchez, and Pasko 2015). In fact, those two techniques were applied to geometric and attribute components of the heterogeneous objects separately.

A natural idea is to develop a unifying method allowing to deal with blending and metamorphosis between heterogeneous objects without separately dealing with their geometric and attribute components. As we will show in this chapter, the STB method, used for geometry transformation, can be naturally combined with the space-time transfinite interpolation (STTI) used for attribute (e.g. colour) transformations. This newly developed HSTB method allows geometry and attribute transformations to be interconnected and happen simultaneously in an higher dimensional specific STB space.

We then show that all the specific 2D metamorphosis techniques introduced in the previous chapter can be extended for the HSTB method to work efficiently in 3D for heterogeneous objects defined by distance-based functions that are at least C^0 or C^1 continuous. Finally, as proof of the concept, we show how how the developed method can be applied to deal with dynamic (time-variant) textured objects in the context of such artistic application as '4D Cubism' that was described in (Corker-Marin, Pasko, and Adzhiev 2018) and was originally implemented for volumetric geometric shapes without attributes.

6.1 Basic Algorithm for Computing 3D Heterogeneous Space-Time Blending (HSTB)

In this section we describe the basic HSTB algorithm. Whilst it is universal in its essence with respect to the attributes of different nature, we will illustrate our approach using such an important and representative attribute as colour. Technically, this means that a higher-dimensional specific STB space in which the geometry

Part of this chapter was published in (Tereshin, Anderson, et al. 2020).

transformation is executed is associated with a 4D colour space with an added time dimension.

Various attributes can be defined for initial and target objects O_1 and O_2 that can be used in the HSTB method. The algorithmic implementation of the method can also slightly varies as attribute transformation can be computed either simultaneously with geometry transformation in a higher dimensional STB space or separately after it. To describe the steps of the basic algorithm, for simplicity we consider colour as an attribute defined for initial and target objects O_1 and O_2.Our basic algorithm approach is formulated as follows:

1. The definition of initial and target heterogeneous objects (Fig. 6.1, frame 1) as hypervolumes using some distance-based functions $F_{DF_1}(\boldsymbol{p}, A_1)$ and $F_{DF_2}(\boldsymbol{p}, A_2)$ with attribute sets A_1 and A_2. We split them into two parts: geometric $F_{DF}(\boldsymbol{p})$ and attribute $A(a_1^*, a_2^*, ..., a_n^*)$, where $\boldsymbol{p}$ is a point in n-dimensional Euclidean space. The geometric part can be defined using an SDF function:

$$F_{DF}(\boldsymbol{p}) = \begin{cases} d(\boldsymbol{p}, \partial G) & \text{if} \quad \boldsymbol{p} \in G \\ -d(\boldsymbol{p}, \partial G) & \text{otherwise} \end{cases}$$

   where $F_{DF}(\boldsymbol{p})$ corresponds to SDF function, $d(\boldsymbol{p}, \partial G)$ is the minimal signed Euclidean distance between point $\boldsymbol{p}$ and the surface ∂G of the object. Alternatively, the geometric part can be defined as an HFRep function:

$$F_{DF}(\boldsymbol{p}) = (F_{st} \circ F_{FRep})(\boldsymbol{p}) \cdot F_{sm}(\boldsymbol{p}, \partial G)$$

   where $F_{DF}(\boldsymbol{p})$ corresponds to HFRep function, $(F_{st} \circ F_{FRep})(\boldsymbol{p})$ is the composition of the step-function $F_{st}(\cdot)$ and FRep function $F_{FRep}(\boldsymbol{p})$ that provides sign to the computed smoothed unsigned distance field $F_{sm}(\boldsymbol{p}, \partial G) = (F_I \circ F_{DF})(\boldsymbol{p}, \partial G)$. Here F_I is a spline-based interpolation function. Both objects can also be a combination of different DF types, e.g. SDFs or HFReps.

2. The use of attribute functions (e.g. procedural) A_1 and A_2 to define attribute (colour) distribution according to the defining functions. We consider that attribute functions A_1 and A_2 for both objects O_1 and O_2 are defined as colour fields $A_1(a_1^*, ..., a_n^*) = c_1(r_1, g_1, b_1)$ and $A_2(a_1^*, ..., a_m^*) = c_2(r_2, g_2, b_2)$. They are defined for each point of the objects O_1 and O_2 with the defining functions $F_{DF_1}(\boldsymbol{p}) \geq 0$ and $F_{DF_2}(\boldsymbol{p}) \geq 0$.

3. Before applying STB method to both objects O_1 and O_2, we can use STB automatic control described in the previous section for setting up STB controlling coefficients a_0, a_1, a_2 and a_3.

4. To apply STB to both objects O_1 and O_2, we raise their defining functions to the higher dimension and make them dependent on time t, applying an additional STB operation defined by equations (5.4) to each object to obtain a smooth transition between them (smoothing half-cylinders technique, equations (6.7)):

$$F_b(f_{1_t}, f_{2_t}, f_{3_t}) = f_{1_t}(\boldsymbol{p}, t) \cap f_{2_t}(\boldsymbol{p}, t) + a_0 disp_b(f_{1_t}, f_{2_t}, f_{3_t}, a_1, a_2, a_3)$$

   where $f_{1_t}(\boldsymbol{p}, t)$ is the DF function raised to a higher-dimension, defining either object O_1 or object O_2, $f_{2_t}(\boldsymbol{p}, t)$ is the function defining a smoothing object

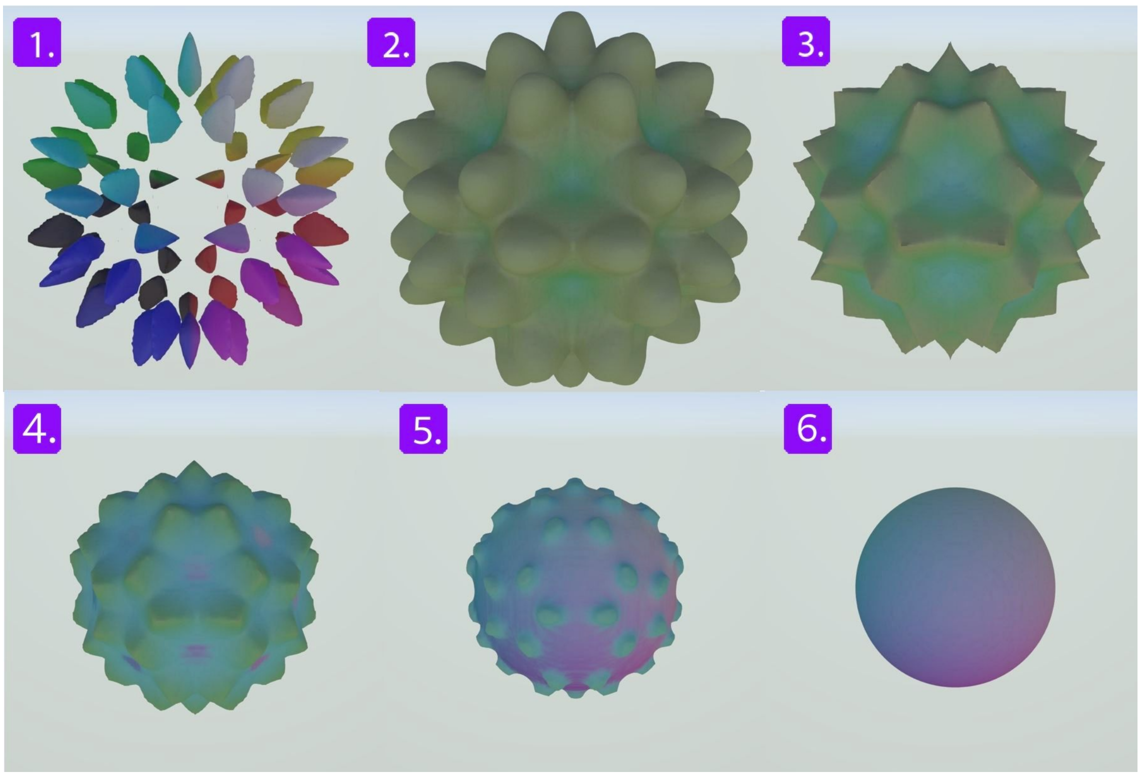


Figure 6.1: *The heterogeneous STB metamorphosis between 'Barth Desic' surface and sphere that was applied with manual control. Supplementary video: figure6.1.mpg*

as the subtraction of the negative half-space along time-axis t, $f_{3_t}(\boldsymbol{p}, t)$ is the bounding function restricting the area of the STB bounding intersection ($\cap$) and P_c is the position of the cutting plane on time-axis t. At this step we can manually define which function for the bounding solid to use (two half-planes, truncated pyramid or truncated cone).

5. For each point in which blending function $F_b(f_{1_t}, f_{2_t}, f_{3_t}) \geq 0$ and defining functions $f_{1_t} < 0, f_{2_t} < 0$ of the objects, in the general case any operation over attributes can be applied. Here we apply STTI defined by equations (5.6)-(5.7) that is also executed in a higher dimension. Colours $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ can be defined in RGB or HSV colour spaces. If HSV colour space is used, we need to find the shortest path between two 'hue' values, and linearly interpolate between them, weighted by w_2, (equation (5.6)) while using the STTI approach for 'saturation' and 'value' values. Finally, STTI is applied and the resulting colour c is converted back to RGB colour space (see Fig. 6.9, B).

6.2 Automatic Control for The 3D Space-Time Blending

As we have already discussed in the previous chapter, the space-time blending (STB) method has several drawbacks. One of them is the manual control of the method, particularly, the definition of STB coefficients $a_i, i = 0, .., 3$ presented in system of equations (5.4). Another one is an non-smooth shape transformation. Finally,

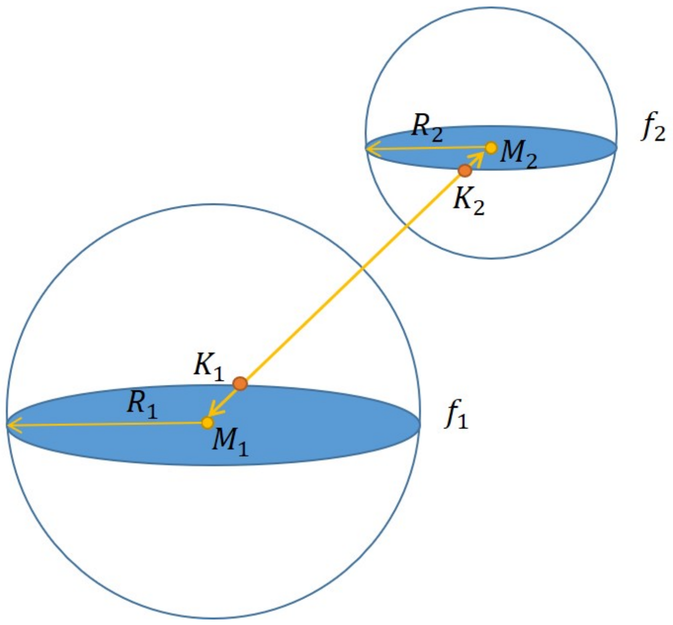


Figure 6.2: *Geometrical scheme for estimating the coefficients $a_1 = a_2$, where the initial shape is defined by function f_1 and the target shape is defined by the function f_2.*

the STB method could produce inflated results. To solve these problems for 2D applications, we have introduced automatic control for STB coefficients a_i, smoothing half-cylinders operation and new bounding solids. In this section we extend introduced techniques to 3D.

6.2.1 Automatic Control of STB Coefficients

In this subsection we extend to 3D the automatic control for STB coefficients a_1 and a_2, which we have already discusses in subsection 5.3.2. Let us assume that we have initial and target objects O_1 and O_2 defined by distance functions $f_1(\boldsymbol{p})$ and $f_2(\boldsymbol{p})$ (e.g. HFReps, SDFs etc.). Let us also assume for simplicity that initial and target objects are defined as a circumscribed spheres around them. The radii of the circumscribed spheres R_1 and R_2, and their centre coordinates M_1 and M_2 are known.

The proposed estimation for the coefficients a_i also relies on a blending equation (Pasko, Pasko, and Kunii 2005) without time dependence that in 3D case can be written as follows:

$$F_{blend}(f_1, f_2) = f_1(\boldsymbol{p}) + f_2(\boldsymbol{p}) + \sqrt{f_1^2 + f_2^2} + \frac{a_0}{1 + d_{r_1}(f_1, f_2)} \tag{6.1}$$

$$d_{r_1}(f_1, f_2) = \sqrt{\left(\frac{f_1(\boldsymbol{p})}{a_{1,2}}\right)^2 + \left(\frac{f_2(\boldsymbol{p})}{a_{1,2}}\right)^2}$$

where $f_1(\boldsymbol{p})$ and $f_2(\boldsymbol{p})$ are functions with distance property and $d_{r_1}(f_1, f_2)$ is a generalised distance between the initial and target shapes represented by functions $f_1(\boldsymbol{p})$ and $f_2(\boldsymbol{p})$.

The estimation process starts from finding the values for coefficients a_1 and a_2. Let us consider that $a_1 = a_2 = a_{1,2}$ that corresponds to a symmetric blend between initial and target objects O_1 and O_2. As stated in (Pasko, Pasko, and Kunii 2005), coefficients $a_1 > 0$ and $a_2 > 0$ are proportional to the algebraic distance between the blending surface and the original surfaces of the objects O_1 and O_2 defined by functions $f_1(\boldsymbol{p})$ and $f_2(\boldsymbol{p})$.

To estimate coefficient $a_{1,2}$ we suggest the following algorithm: as it was discussed in the previous chapter, the generalised distance $d^2_{r_1}(f_1, f_2)$, used in equation (6.1), is equal to the quadratic distance $||s||^2$ (see Fig. 6.2) between two objects O_1 and O_2 that can be defined as the distance between two points K_1 and K_2:

$$||s||^2 = ||K_2 - K_1||^2 \tag{6.2}$$

To compute this distance we need to solve the following system of equations taking into account that initial and target objects are assumed to be defined as spheres with radii R_1 and R_2 and centres M_1 and M_2 (see Fig. 6.2):

$$\begin{cases} f_{sp_1}(\boldsymbol{p}) = ||\boldsymbol{p} - M_1||^2 - R_1^2 \\ f_{sp_2}(\boldsymbol{p}) = ||\boldsymbol{p} - M_2||^2 - R_2^2 \\ \frac{p_x - M_{1_x}}{a} = \frac{p_y - M_{1_y}}{b} = \frac{p_z - M_{1_z}}{c} \end{cases} \tag{6.3}$$

where $a = M_{1_x} - M_{2_x}$, $b = M_{1_y} - M_{2_y}$ and $c = M_{1_z} - M_{2_z}$, $\boldsymbol{p}$ corresponds to either K_1 or K_2 and we solve this system for point $\boldsymbol{p}$. The solution of this system for points K_1 and K_2 can be written as follows:

$$\boldsymbol{K}_i = \left(\frac{aR_i}{\sqrt{a^2+b^2+c^2}} + M_{i_x}, \frac{bR_i}{\sqrt{a^2+b^2+c^2}} + M_{i_y}, \frac{cR_i}{\sqrt{a^2+b^2+c^2}} + M_{i_z} \right) \quad i = 1, 2 \tag{6.4}$$

Finally, we obtain the final estimation for $a_1 = a_2 = a_{1,2}$:

$$a_{1,2} = \sqrt{\frac{f_1(K_2)^2 + f_2(K_1)^2}{||s||^2}} \tag{6.5}$$

In Fig. 6.3 we show the result of applying of the proposed method for automatic control of coefficients a_1 and a_2 that according to the proposed method of estimation were equal to 0.41. Coefficient a_3 was set to one and coefficient a_0 was set to 0.3. The resulting blending shape is less inflated and better controlled.

6.2.2 Smoothing Half-Cylinders

As it was stated in subsection 5.3.1, one of the STB drawbacks is the presence of fast transitions in the time interval $t \in [0, 1]$, where the most significant shape transformation happens. In that subsection, we have discussed the causes of these fast transitions. To compensate for this drawback, we have introduced a smoothing half-cylinders technique for 2D case that can be extended to 3D case in a straight forward manner. The smoothing procedure is based on the bounded blending intersection operation that in 3D can be defined as follows:

$$\begin{aligned} F_b(f_{1_t}, f_{2_t}, f_{3_t}) &= F_\cap(f_{1_t}, f_{2_t}) + a_0 \cdot disp_b(d_r(f_{1_t}, f_{2_t}, f_{3_t})); \\ F_\cap(f_{1_t}, f_{2_t}) &= f_{1_t}(\boldsymbol{p}, t) + f_{2_t}(\boldsymbol{p}, t) - \sqrt{f_{1_t}(\boldsymbol{p}, t)^2 + f_{2_t}(\boldsymbol{p}, t)^2}; \end{aligned} \tag{6.6}$$

where $\boldsymbol{p}$ is a 3D point, $F_b(f_{1_t}, f_{2_t}, f_{3_t})$ is the resulting blending function defining a smoothed half-cylinder, $disp_b(d_r(f_{1_t}, f_{2_t}, f_{3_t}))$ is the displacement defined using equations (5.3)-(5.4), and $F_\cap(f_{1_t}, f_{2_t})$ is a FRep set-theoretic intersection operation.

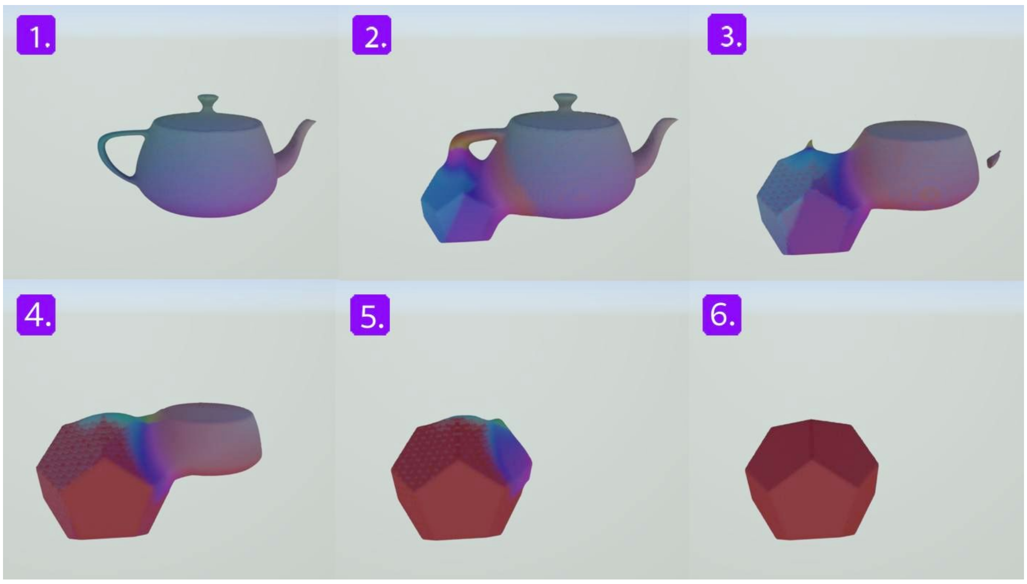


Figure 6.3: *The HSTB result obtained with automatic control for the a_1 and a_2 coefficients. Supplementary video: figure6.3.mpg*

The smoothing process for initial and target objects O_1 and O_2 can be described as follows:

$$\begin{aligned} &O_1: \ f_{1_t}(\boldsymbol{p},t) = f_{DF_1}(\boldsymbol{p},t); \ f_{2_t}(\boldsymbol{p},t) = -t; \ f_{3_t}(\boldsymbol{p},t) = t + P_c, \ t \geq -P_c; \\ &O_2: \ f_{1_t}(\boldsymbol{p},t) = f_{DF_2}(\boldsymbol{p},t); \ f_{2_t}(\boldsymbol{p},t) = t - 1; \ f_{3_t}(\boldsymbol{p},t) = P_c - t, \ t \leq P_c; \end{aligned} \tag{6.7}$$

where $f_{1_t}(\boldsymbol{p},t)$ is the SDF or HFRep function raised into four-dimensional space defining either the initial object O_1 or the target object O_2, $f_{2_t}(\boldsymbol{p},t)$ is the function defining a smoothing object as the subtraction of the negative half-space along time-axis t, $f_{3_t}(\boldsymbol{p},t)$ is the bounding function restricting the area where the STB bounding intersection happens, and P_c is the position of the cutting plane on the time-axis t. These functions are substituted in equation (6.6) for obtaining the smoothed result. For smoothing both the initial object O_1 as well as the target object O_2, we need to manually set the coefficients a_0, a_1, a_2 and a_3.

6.2.3 Truncated Cone and Truncated Pyramid as a Bounding Solid

In subsection 5.3.3 we have introduced two new bounding solids for the additional control of the undesired inflation caused by the lack of the STB control. These bounding solids are truncated cone and truncated pyramid.

First, let us redefine bounding solid in the form of truncated cone in 3D space. Let us assume that we have already obtained two bounding spheres with radii R_1 and R_2 for the initial and target objects O_1 and O_2. The cross section of each sphere will be circles with radii R_1 and R_2 that will serve as the basis for the top and bottom bases of the truncated cone. To avoid the blending shape inflation, we specify that the entire metamorphosis process happens within the new bounding solid defined by function $f_{3_t}(\boldsymbol{p},t)$.

To obtain the defining function $f_{3_t}(\boldsymbol{p},t)_{cone}$ for the truncated cone, a simple linear interpolation between the centres and radii of the circles is applied, where the

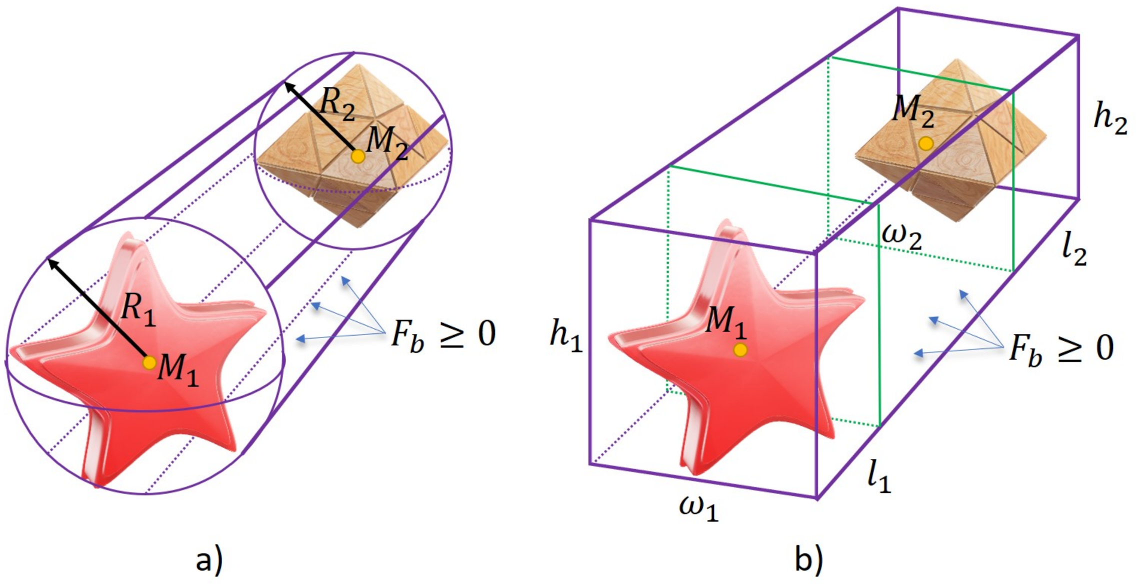


Figure 6.4: *New bounding solids for additional control of the STB metamorphosis. a) Truncated cone; b) Truncated pyramid.*

parameter k is defined in the interval $k \in [0, 1]$, which is dependent on time t:

$$\begin{gathered} f_{3_t}(\boldsymbol{p}, t)_{cone} = R_m^2 - (p_x - \widehat{X}_m)^2 - (p_y - \widehat{Y}_m)^2 - (p_z - \widehat{Z}_m)^2 \\ R_m = R_1 + (R_2 - R_1)k, \quad \widehat{X}_m = x_{c_1} + (x_{c_2} - x_{c_1})k, \\ \widehat{Y}_m = y_{c_1} + (y_{c_2} - y_{c_1})k, \quad \widehat{Z}_m = z_{c_1} + (z_{c_2} - z_{c_1})k, \\ k = \frac{t - t_1}{t_2 - t_1}, \quad t_1 = -10; \quad t_2 = 10. \end{gathered} \tag{6.8}$$

Here $M_1(x_{c_1}, y_{c_1}, z_{c_1})$ and $M_2(x_{c_2}, y_{c_2}, z_{c_2})$ are the centres of the circles with radii R_1 and R_2 obtained as a cross sections of the bounding spheres generated for initial and target object O_1 and O_2, t_1 and t_2 are the minimum and the maximum values of the interval time t. We use the values recommended for t_1 and t_2 from the STB paper (Pasko, Pasko, and Kunii 2005).

To redefine the bounding solid in the form of the truncated pyramid we will apply the same steps as we have done earlier in this subsection. Let us assume that we have obtained two rectangular bounding boxes around initial and target object O_1 and O_2 with widths l_1 and l_2, heights h_1 and h_2 and depths d_1 and d_2. The cross sections of these bounding boxes will be rectangles with centres in points $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$.

We define each base of the pyramid as a result of the FRep set-theoretic intersection operation in 3D space, and linear interpolation between the widths l_1, l_2, the heights h_1, h_2, the depths d_1, d_2 and the centres of the rectangles. Then the bounding solid in the form of the truncated pyramid is defined as follows:

$$\begin{gathered} f_{3_t}(\boldsymbol{p}, t)_{pyramid} = (W_m - |p_x - \widehat{X}_m|) \cap (H_m - |p_y - \widehat{Y}_m|) \cap (D_m - |p_z - \widehat{Z}_m|) \\ W_m = l_1 + (l_2 - l_1)k, \quad H_m = h_1 + (h_2 - h_1)k, \quad D_m = d_1 + (d_2 - d_1)k \\ \widehat{X}_m = x_{r_1} + (x_{r_2} - x_{r_1})k, \quad \widehat{Y}_m = y_{r_1} + (y_{r_2} - y_{r_1})k, \\ \widehat{Z}_m = z_{r_1} + (z_{r_2} - z_{r_1})k, \quad k = \frac{t - t_1}{t_2 - t_1}, \quad t_1 = -10; \quad t_2 = 10. \end{gathered} \tag{6.9}$$

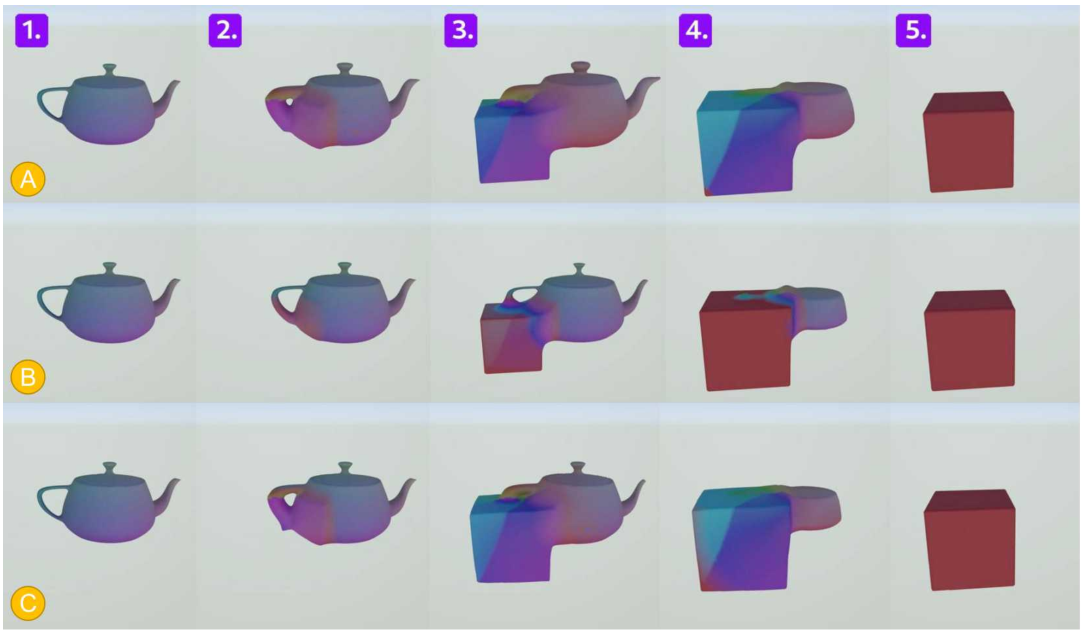


Figure 6.5: *In this picture we show three sequences of frames to illustrate how two half-planes, truncated pyramid and truncated cone bounding solids control the blending shape. A) The bounding solid is is defined as an intersection of two half-planes* $(t+t_c)$ *and* (t_c-t)*. B) The bounding solid is defined as a truncated pyramid, equation (6.9). C) The bounding solid is defined as a truncated cone, equation (6.8). Supplementary videos: figure6.5(a).mpg, figure6.5(b).mpg, figure6.5(c).mpg*

where $M_1(x_{r_1}, y_{r_1}, z_{r_1})$ and $M_1(x_{r_1}, y_{r_1}, z_{r_1})$ are the centres of the rectangular basis.

To define a new function for the bounding solid as either a truncated cone or a truncated pyramid we need to use the FRep set-theoretic intersection operation $\cap$ between either the cone defined by $F_{solid}(\boldsymbol{p}, t) = f_{3_t}(\boldsymbol{p}, t)_{cone}$ or the pyramid defined by $F_{solid}(\boldsymbol{p}, t) = f_{3_t}(\boldsymbol{p}, t)_{pyramid}$ with the two time-cutting planes $(t+C)$ and $(C-t)$

$$f_{3_t}(\boldsymbol{p}, t) = F_{solid}(\boldsymbol{p}, t) \cap (t+C) \cap (C-t) \tag{6.10}$$

where C is some constant.

In Fig. 6.5 we present three sequences of frames that are obtained using HSTB computed with three different bounding solids. The initial and target objects 'utah teapot' and 'box' were initially defined as BReps and then converted to SDFs. The colour was interpolated using linear interpolation in HSV colour space weighted by the values of the defining distance function for the second object ('box').

In Fig. 6.5, (A), we show the STB result that was computed with a bounding solid defined as two half-planes. If we compare it with the results obtained using truncated pyramid (see Fig. 6.5, B) or truncated cone (see Fig. 6.5, C) as a bounding solid, we can see that two-half planes produces the most inflated result. The best shape control and colour interpolation is achieved using truncated pyramid. In addition to it, it is hard to manually adjust the STB coefficients a_i when the truncated cone is set up as a bounding solid.

6.3 HSTB: Algorithm and Implementation

In this section we discuss an algorithmic solutions for space-time blending as well as heterogeneous space-time blending with automatic control that were implemented

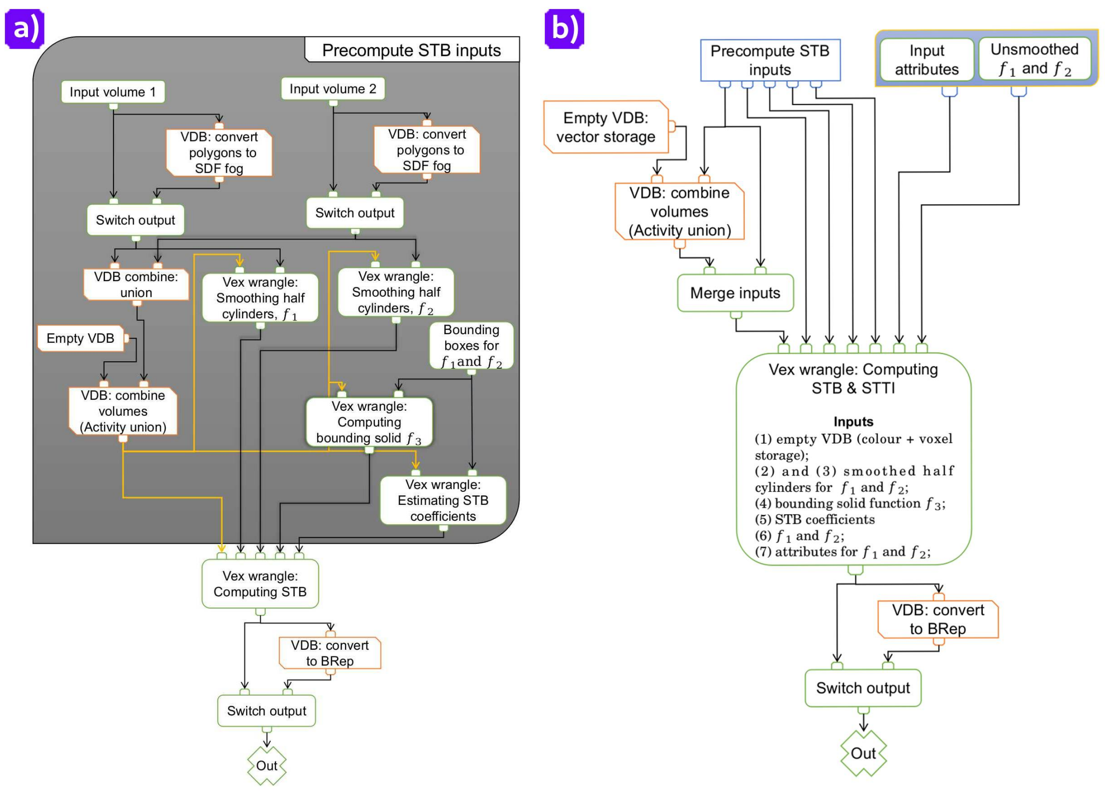

a)
Precompute STB inputs
Input volume 1
Input volume 2
VDB: convert polygons to SDF fog
VDB: convert polygons to SDF fog
Switch output
Switch output
VDB combine: union
Vex wrangle: Smoothing half cylinders, f_1
Vex wrangle: Smoothing half cylinders, f_2
Empty VDB
Bounding boxes for f_1 and f_2
VDB: combine volumes (Activity union)
Vex wrangle: Computing bounding solid f_3
Vex wrangle: Estimating STB coefficients
Vex wrangle: Computing STB
VDB: convert to BRep
Switch output
Out
b)
Precompute STB inputs
Input attributes
Unsmoothed f_1 and f_2
Empty VDB: vector storage
VDB: combine volumes (Activity union)
Merge inputs
Vex wrangle: Computing STB & STTI
Inputs
(1) empty VDB (colour + voxel storage);
(2) and (3) smoothed half cylinders for f_1 and f_2;
(4) bounding solid function f_3;
(5) STB coefficients
(6) f_1 and f_2;
(7) attributes for f_1 and f_2;
VDB: convert to BRep
Switch output
Out

Figure 6.6: *The algorithmic implementation of the heterogeneous space-time blending. a) the algorithmic implementation of the STB method for handling geometric transformations; b) the algorithmic implementation of the heterogeneous STB for handling both geometric and attribute transformations simultaneously.*

in *SideFX Houdini* (Side Effects Software 2020). We provide a detailed description of our implementation with some examples.

6.3.1 Space-Time Blending Algorithm

In this chapter we have extended the automatic control for the STB method to 3D space. The heterogeneous STB method with semi-automatic control consists of several parts, namely, techniques for automatic estimation of STB coefficients a_1, a_2, the STB computation for geometry transformation and STTI computation for attribute transformation. In this subsection we describe the algorithmic solution for STB computation including its automatic control and new bounding solids.

In Fig. 6.6 we present an algorithmic diagrams for the heterogeneous STB method implementation in *SideFX Hodini*. We split it into two parts. In Fig. 6.6 (a) we presented the algorithmic solution for the STB method without taking into account attribute properties of the input objects O_1 and O_2 defined by the functions $f_1(\boldsymbol{p})$ and $f_2(\boldsymbol{p})$. We also highlight the part of the algorithm ('Precompute STB inputs') that will remain the same for both STB and heterogeneous STB implementations.

The input objects O_1 and O_2 defined by the functions $f_1(\boldsymbol{p})$ and $f_2(\boldsymbol{p})$ can be of arbitrary topology and defined as HReps or SDFs, or BReps. If the input objects are defined using BRep representation we can convert them to VDB SDF fog. Otherwise, it is assumed that input objects are defined as VDB fog volumes. The main advantage of the VDB fog volumes over level-set volumes is that it could store any

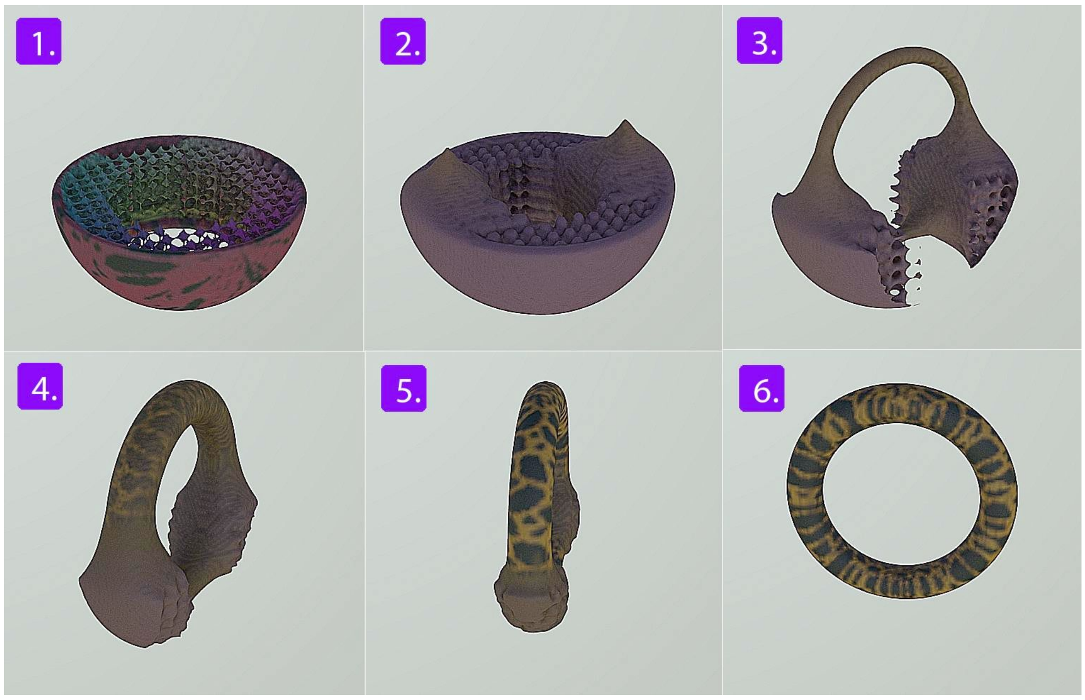


Figure 6.7: *In this picture we show the result of HSTB applied to two HFRep objects. The initial multi-material object consists of two materials with defined microstructures in its interior and the target object 'torus' has only one texture attribute. Supplementary video: figure6.7.mpg*

non-distance data that is crucial in case of STB as it does not preserve distances. Then, we have to do several computational pre-steps before computing the STB method.

On the first step, we need to apply smoothing the half-cylinders technique to both input objects O_1 and O_2 defined by the functions $f_1(\boldsymbol{p})$ and $f_2(\boldsymbol{p})$, and raise them into 4D space, where the fourth coordinate will be time t. This operation defined by equations (6.7) is implemented in two *VEX Volume Wrangle VOPs*, and it solves the STB problem related to fast transitions during metamorphosis. Then, we need to pre-compute the STB controlling coefficients $a_i, i = 1, 2$ in *VEX Volume Wrangle VOP* using geometric information of the bounding boxes defined as the spheres using introduced techniques in subsection 6.2.1 Finally, it is required to compute in the *VEX Volume Wrangle VOP* one of the bounding solids discussed in subsection 6.2.3 that are defined by the function f_3.

These steps cover the 'Precompute STB inputs' stage. The generated parameters are used as an input for the STB computation in VEX volume wrangle using equations (5.4)-(5.3). The output of this STB implementation will be either VDB non-distance based fog or BRep surface.

6.3.2 Heterogeneous Space-Time Blending Algorithm

The algorithmic implementation of the automatically controlled heterogeneous space-time blending (HSTB) is based on the STB algorithm that can be seen in Fig. 6.6 (a). According to the HSTB algorithm shown in Fig. 6.6 (b), we need to repeat all steps of the 'Precompute STB inputs' stage. We also need to specify attributes for each input object O_1 and O_2 defined by HFRep or SDF functions $f_1(\boldsymbol{p})$ and $f_2(\boldsymbol{p})$.

The attributes can be defined procedurally using *VEX volume wrangle VOP* and stored in a separate VDB fog volume, where each voxel stores a vector of three float values that correspond to colours. Alternatively, attributes can be defined as a simple colour distribution with 'colour node'. Generally, the input objects O_1 and O_2 can be represented as a merged VDB volume that consists of one fog voxel grid with distance field values and one fog voxel grid that stores the attribute values.

We define an empty VDB volume that consists of two VDB fog grids, where we will store the results of the computation of STB and STTI methods. First, we compute the STB function F_b value for the current voxel according to equations (5.4)-(5.3). Then, if $F_b > 0$, we apply STTI method defined by the equations (5.6) - (5.7) to interpolate new colours in the added by blending function volume. The attribute (colour) interpolation can be made in two colour spaces HSV and RGB. Both STB and STTI methods are implemented in a single VEX volume wrangle. The output of the HSTB node can be either a VDB fog volume that consists of two VDB voxel grids or a BRep object with attributes transferred from the VDB volume to its surface.

In Fig. 6.7 we show a sequence of frames that were obtained using HSTB. The HSTB method was computed for the initial and target HFRep objects. The initial HFRep object is obtained using several set-theoretic operations. First, a cylinder was subtracted from the centre of the sphere. Then, using Algorithm 4.18 a shelled sphere was obtained that was further combined with generated microstructures according to the Algorithm 4.19. Finally, two materials were assigned to the shell of the object and to its interior. In this example, the target HFRep 'torus' object rotates around its vertical axis.

6.4 HSTB: '4D Cubism'

In this work, we broaden the concept of '4D Cubism' (Corker-Marin, Pasko, and Adzhiev 2018), an artistic application that provides tools for creating artistic shapes in a cubist style employing the STB method that could only deal with geometric metamorphosis without attributes, by introducing attribute transformations and the concept of heterogeneous STB for metamorphosis between volumetric objects. We also apply this concept to the objects defined by different types of DF based representations, e.g. SDFs and HFReps. All examples are implemented in *SideFX Houdini* using the OpenVDB library for handling both attributes and geometry transformations, rendered using an Intel Xeon E5-1650 3.20 GHz PC with 32 Gb of RAM.

6.4.1 HSTB: Implementation

In the '4D Cubism' application the input objects are defined as BReps. The framework described in (Corker-Marin, Pasko, and Adzhiev 2018), is based on shape faceting, the conversion of the obtained after faceting BRep objects to SDFs, and computation of the space-time blending (STB) to perform a smooth transformation between two input SDF objects.

The aim of the shape-faceting step is to subdivide the shape of the input BRep object into facets. A facet is considered as a solid piece of geometry obtained by the set-theoretic intersection between some solid primitive (e.g. solid tetrahedron) with the initial shape of the object converted to a solid. The shape faceting is applied using octree data structure that subdivides the shape of the initial object for further assignment of the facets. It is also possible to specify different unique

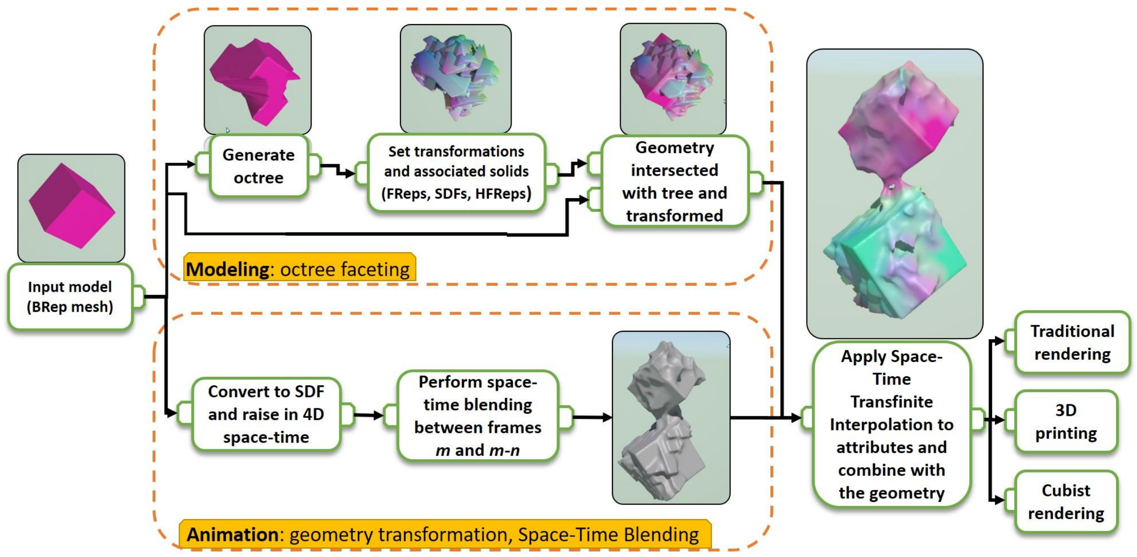

Generate octree
Set transformations and associated solids (FReps, SDFs, HFReps)
Geometry intersected with tree and transformed
Input model (BRep mesh)
Modeling: octree faceting
Convert to SDF and raise in 4D space-time
Perform space-time blending between frames m and m-n
Animation: geometry transformation, Space-Time Blending
Apply Space-Time Transfinite Interpolation to attributes and combine with the geometry
Traditional rendering
3D printing
Cubist rendering

Figure 6.8: *The algorithmic implementation of the 4D Cubism with heterogeneous attribute transformations.*

transformations to each facet during this stage. Finally, the obtained distorted shapes are converted to SDFs and STB is computed to obtain smooth blending between input shapes.

In Fig. 6.8 we show an extended algorithm for the '4D Cubism' application that also cover not only shape transformation, but also attribute transformations. At the faceting step different HFRep objects can be used as facets. For each facet we can assign different attributes (e.g. colours, textures). The attributes for the input undistorted shape can be defined as 2D textures, or after their conversion to SDFs they can be set up as 3D textures or procedural textures. After conversion of the input shape to SDFs we can also define attributes in the form of HFRep microstructures incorporated in their interior (see subsection 4.6.2 for the algorithmic implementation).

Instead of the STB method we introduce a heterogeneous STB method with a possible semi-automatic control of STB parameters and additional shape control. The heterogeneous STB method is based on STB and STTI methods coupled together and their implementation in *SideFX Houdini* was discussed in subsection 6.3.2.

6.4.2 Application and Results

We demonstrate our approach using three examples. The first example (Fig. 6.9) applies heterogeneous STB to a heterogeneous oscillating SDF 'cube' object and an HFRep 'heart' object. In the basic algorithm, we first define the initial (SDF 'cube') and target (HFRep 'heart') objects. We then define the colour attributes for both objects.

Next we raise both defining functions for the 'cube' and the 'heart' objects to a higher STB dimension using equations (6.7). Finally, we apply the introduced heterogeneous STB to both objects and compute both in-between geometry and colour attribute transformations simultaneously using RGB (see Fig. 6.9, a) and HSV (see Fig. 6.9, b) colour spaces We implemented interpolation in HSV using linear interpolation and STTI interpolation. As it can be seen in Fig. 6.9, STTI provides a smoother colour transition than linear interpolation, and using an HSV

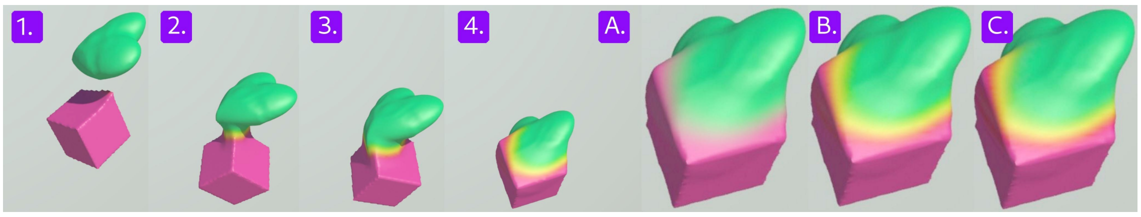


Figure 6.9: *Result of applying heterogeneous STB to an SDF cube object and an HFRep heart object. The result of applying (A) STTI for RGB colour interpolation; (B) STTI for HSV colour interpolation; (C) linear HSV colour interpolation. Supplementary video: figure6.9(hsv_linear).mpg, figure6.9(hsv_stti).mpg, figure6.9(rgb_stti).mpg*

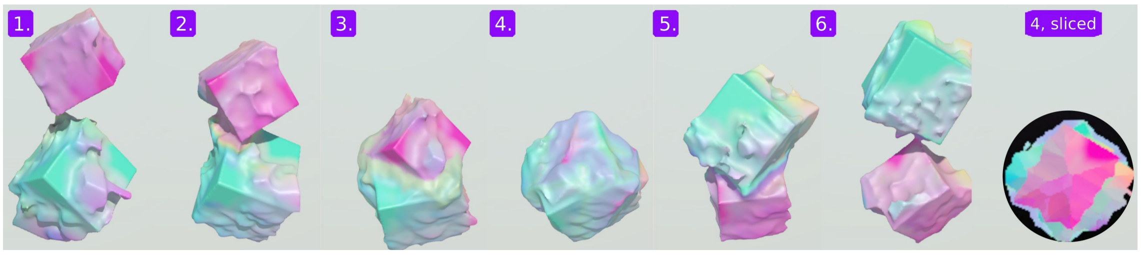


Figure 6.10: *Frame sequence of heterogeneous STB applied to two heterogeneous objects with multiple features defined by HFRep and SDF. Supplementary video: figure6.10(hsv_linear).mpg, figure6.10(rgb_stti).mpg, figure6.10(hsv_stti).mpg, figure6.10(hsv_stti_slice).mpg*

colour space results in a smoother transition between colours comparing to an RGB colour space.

In Fig. 6.10 we show a more complex example of two oscillating heterogeneous objects with cubist coloured features demonstrating that HFRep objects can easily be combined with SDF objects. First, two polygonal cubes are converted to OpenVDB objects. Then SDF functions for both cubes are obtained. To specify where coloured features will be added, both cubes are subdivided using an octree. Some of the features are defined using HFRep, some using SDF representations. Next we specify colour attributes for these cubes and assigned features. At the third step, we combine functions that define features with functions specifying the base cubes, using set-theoretic operations, and raise the resulting functions to a 4D space using equations (6.7). Finally, we compute simultaneously STB and STTI (HSV colour space) methods, and then map the STTI result computed in a separate voxel grid to the obtained geometry. Note that operations on SDFs and HFReps cannot preserve attributes in *SideFX Houdini*. Therefore, they are computed using separate vectorised OpenVDB voxel grids.

In Fig. 6.11 we show a more sophisticated example of applying the proposed HSTB technique in the context of '4D Cubism'. We have applied the introduced HSTB concept to a model of a walking person. As we have stated in the basic algorithm (section 6.1), step five, we can use any operation for handling attribute transformations. In this example we compute the attribute transformation in an HSV colour space using STTI interpolation.

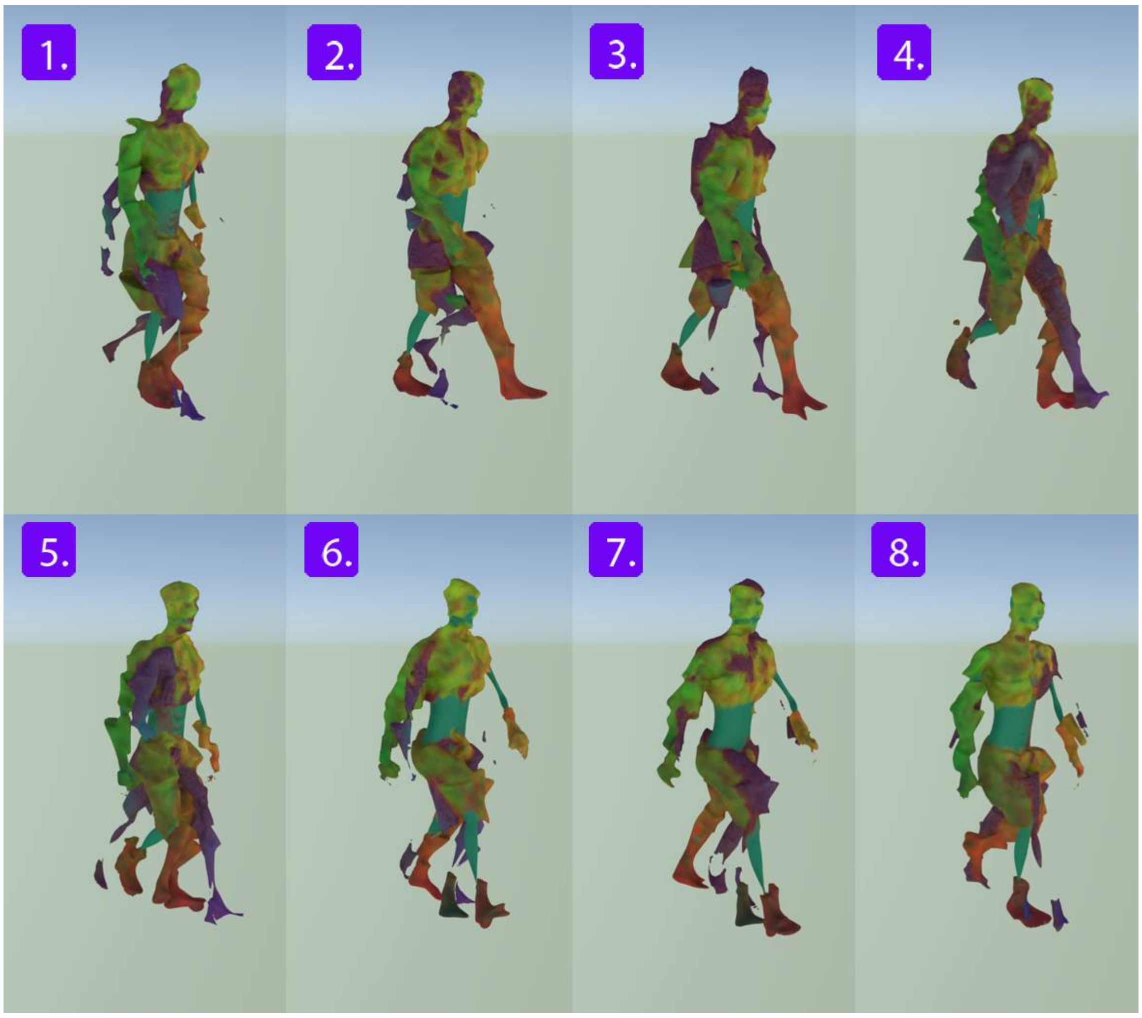


Figure 6.11: *A sequence of frames that shows the result of applying heterogeneous STB to two heterogeneous objects in the form of a walking figure with multiple features defined by HFRep and SDF. A linear colour interpolation in HSV colour space was applied to handle the attribute transformations. Supplementary video: figure6.11.mpg*

6.4.3 Conclusions

In this chapter we have presented the concept of heterogeneous space-time blending (HSTB) based on STB for handling geometry transformation and STTI for handling attribute transformations. In the general case, instead of STTI, any method that is suitable for attribute transformation can be used. We have extended several STB techniques introduced in the previous chapter for additional shape control during metamorphosis, namely smoothing half-cylinders, automatic control for STB coefficients $a_1 = a_2$ and two new bounding solids (truncated cone and truncated pyramid).

We have also presented the basic algorithm for implementing HSTB that we further applied to dynamic objects ('4D Cubism' (Corker-Marin, Pasko, and Adzhiev 2018)). We have suggested to compute attribute interpolation in HSV colour space as colour transitions in this system are natural for a perception of human eyes. We have implemented the basic algorithm in *SideFX Houdini* and provided a detailed description of our implementation with corresponding examples.

We have extended '4D Cubism' application with volumetric attributes in the form of colour distributions that are parameterised by the position of the point or by the bounding box values of the voxelised geometry. For representing heterogeneous objects, we have used SDFs implemented inside *OpenVDB* and HFRep custom node. The implementation of the HFRep node was discussed in subsection 4.6.2. The HSTB method was presented in Eurographics short paper (Tereshin, Anderson, et al. 2020).

Chapter 7

Conclusions and Future Work

Heterogeneous multi-material volumetric modelling is an emerging and rapidly developing field.The particular focus of this research was on a systematic and rigorous exploration of the fundamentals of the topic, namely on the mathematically substantiated and algorithmically developed representational schemes for heterogeneous objects geometry and attributes.

The research has been conducted in accordance with the recognised scientific methodology, and the thesis structure reflects the following stages of a well-established research process: motivation with identification of the research questions (chapter 1), exploration of the state of the art in the field with conclusions related to the topic's subject, (chapter 2), theoretical exploration with suggestions and a rigorous substantiation of the topic's fundamentals (chapter 3), developing an algorithmic framework allowing for the efficient practical implementation with introduction of a number of novel computational procedures (chapter 4), development, to prove the concept, of a number of practical methods in the context of particular applications, especially of an artistic nature (chapters 5 and 6), and finally, conclusions with detailed outline of original contributions and some directions for future works (this chapter 7).

There is also an extensive 'References' section as well as the author's list of publications showing that all the main theoretical and practical results of this research work have been peer-reviewed and published/accepted for publication in respectable journals (full-scale papers in 'SIAM Journal of Imaging Sciences', 'Graphical Models') and conferences (short papers in Eurographics 2019 and 2020).

While the boundary representation will remain the main and prevailing instrument for geometric modelling, we believe that the functionally-based representations generalising a well-established implicit modelling approach, are becoming more important in the context of some important modern applications. Hopefully, the hybrid modelling framework allowing conveniently and efficiently deal with heterogeneous multi-material volumetric objects, including those of a time-variant nature, will find its applications.

7.1 Contributions

Overall, we have introduced a theoretical and practical framework for modelling volumetric heterogeneous objects on the basis of a novel unifying functionally-based hybrid representation called HFRep. Let us summarise the specific main contributions of this work:

- A thorough survey of the relevant function-based representations aimed at their classification that allowed to identify four conventional representational schemes related to scalar fields of different kinds, namely function representation (FRep), the adaptively sampled distance function representation (ADF) and the interior distance function representation (IDF). Their advantages and drawbacks have been systematically described. We have suggested a formalisation of the notions of ADFs and IDFs.The requirements for a hybrid representational framework have been formulated.

- The introduction of a mathematically substantiated modelling framework for heterogeneous objects called hybrid function representation (HFRep) that is based on the combination of FRep with one of the three distance-based representations, namely SDF, ADF and IDF to preserve their advantages and compensate for their drawbacks. This novel representational scheme aims to define multi-material heterogeneous objects with their interior structure using a generalisation of the hypervolume model. The formal definition of the HFRep has been formulated. The mathematical properties of the HFRep function have been rigorously described. It was substantiated that the function is at least C^0 or C^1 continuous, smooth and, if it is C^1 continuous, it can be used for defining smoothed attribute distributions. It was shown, the HFRep function can also be made time-variant. The properties of the supported HFRep heterogeneous objects have been systematically described, and the operations over HFRep objects, both those that preserve Euclidean distances and those that do not, were identified. Finally, the basic algorithm for generating HFRep objects in terms of their geometry and attributes has been proposed.

- The introduction of a FIM-based method for the computation of the solution of the eikonal equation on 2D hierarchical grids. The hierarchical FIM (HFIM) method provides at least C^1 continuous unsigned distance field that is restored at each cell of the grid using the PHT-splines. The solution of the eikonal equation is computed using a modified version of the first order Godunov upwind discretisation scheme for computations on the hierarchical grids. The accuracy of the computing the eikonal equation solution was increased by handling T-junctions appeared in the hierarchical grid. The step-by-step algorithm implementing the HFIM method has been suggested. The detailed description covers the used data-structures, the basic quadtree generation, the computation of PHT-splines and the technical implementation of the method.

- The detailed discussion of the methods for defining various attributes in the interior of the 2D and 3D HFRep object. These attributes can be defined either procedurally, or parameterised by the HFRep function that is at least C^1 continuous or on per-voxel basis. It is also possible to define microstructures in the interior of the HFRep object, that is particularly important for the related 3D printing tasks.

- The introduction of a novel method allowing for automatically controlled morphing between two topologically arbitrary 2D shapes with sophisticated textures (raster colour attributes) using the established techniques, namely the space-time blending (STB) coupled with space-time transfinite interpolation (STTI). The method allows for a smooth transition between source and target objects (considered as HFRep objects) by generating in-between shapes and associated textures without setting any correspondences between boundary points or features. The method requires no preprocessing and can be applied in 2D animation when position and topology of source and target objects are significantly different. The basic algorithm for this method has been developed along with several novel techniques solving the identified STB drawbacks. These techniques are half-cylinders STB smoothing, the automatic control for choosing STB coefficients, new bounding solids (truncated cone and truncated pyramid) and affine transformations for better control of the computing STB shape. The method can handle in-place morphing as well as spaced objects morphing.

- The introduction of a novel heterogeneous space-time blending (HSTB) method that can handle holistic heterogeneous objects, rather than their geometric and attribute components in separation, especially in the context of automatically controlled metamorphosis between topologically arbitrary 3D and 4D textured objects with their geometry and attribute transformations executed simultaneously and interconnectedly. The basic algorithm and several techniques, extended to 3D, that solve the identified drawbacks, have been developed.

7.2 Future Work

A lot of research is still required to provide an efficient universal framework for multi-material heterogeneous modelling using the HFRep representation.

It is essential to extend and develop new operations over HFRep objects in the context of different applications, especially related to physical simulation, additive manufacturing and visual effects. It is important to develop set-theoretical operation that are at least C^1 continuous, preserve distance property and are efficient to compute as most of the existing approaches have high complexity of computations.

In technical terms, we aim to develop an efficient HFRep field extrapolation procedure beyond the computational domain that can also be used for extending IDFs in exterior of the object and make IDFs signed. We plan to extend the method for generating IDFs being computed on a tetrahedral mesh. This means voxelising the mesh and defining a band of voxels in exterior of the mesh. Then we can interpolate the IDF values to the centres of the voxels in interior of the voxelised mesh and extrapolate obtained values to the empty voxels outside the voxelised mesh.

One of the interesting directions will be introduction of the multi-material attribute definition in the interior of the volumetric object using the diffusion-based IDFs. IDFs are smooth and are at least C^1 continuous. According to our simple tests with parameterisation of the procedural texture functions by the IDF distances, we were able to obtain very sophisticated textures. Another interesting and promising direction will be the development of the HFRep distance-based volumetric sculpting tool that is suitable for interactive modelling of the multi-material heterogeneous objects.

There are several needed extensions and improvements to be further developed and implemented for the introduced HFIM method. First, it is important to optimise our implementation to increase the efficiency of computations. The GPU-based version of the HFIM algorithm might increase the computational speed for 3D objects. Then, to obtain more accurate resulting unsigned distance field, we need to develop an automatic refinement of the computational hierarchical grid according to the error-based estimation of the already computed solution of the eikonal equation. Finally, there are plans to extend the HFIM method for computing the solution of the eikonal equation on volumetric hierarchical grids, e.g. octrees.

The HSTB method introduced in this work could be expanded in the future by exploring alternative approaches for interpolating attributes between two heterogeneous objects as well as by introducing an automatic feature-based colour segmentation to achieve more sophisticated interpolation between textured objects. In order to achieve further improvements, it will also be beneficial to focus on other areas that were not touched upon in this work, such as optimisation of the colour averaging algorithm or using a quad-tree or similar data structures for acceleration. We are also planning to develop new methods for defining various attributes and their transformations. We would like to broaden the class of attributes that we can deal with using our HSTB approach, e.g. volumetric materials and transparency. Finally, it is essential to develop an algorithm for estimating the STB controlling coefficients in 3D case.

7.3 List of Publications

- Tereshin A., Adzhiev V., Fryazinov O., Pasko A., Hybrid Function Representation with Distance Properties, In: Eurographics 2019 - Short Papers (2019), pp. 17-20, doi: https://doi.org/10.2312/egs.20191004;

- Tereshin A., Adzhiev V., Fryazinov O., Marrington-Reeve F., Pasko A., Automatically Controlled Morphing of 2D Shapes with Textures, In: SIAM Journal on Imaging Sciences, 13, 1, (2020), pp. 78-107, doi: https://doi.org/10.1137/19M1241581;

- Tereshin A., Anderson E., Pasko A., Adzhiev V., Spacce-Time Blending for Heterogeneous Objects, In: Eurographics 2020 - Short Papers (2020), pp. 45-48 , doi: https://doi.org/10.2312/egs.20201014;

- Tereshin A., Pasko A., Fryazinov O., Adzhiev V., Hybrid Function Representation for Heterogeneous Objects, preprint ArXive (2020), 26 pages. Preprint of the paper: https://arxiv.org/abs/2012.15176;

- Tereshin A., Pasko A., Fryazinov O., Adzhiev V., Hybrid Function Representation for Heterogeneous Objects, In: Graphical Models (2021), 114, pp. 101098, doi: https://doi.org/10.1016/j.gmod.2021.101098.

References

Allegre R., Barbier A., Galin E., and Akkouche S. (2004). "A Hybrid Shape Representation for Free-Form Modelling". In: *Proceedings of the Shape Modeling International 2004*. SMI '04. IEEE Computer Society, pp. 7–18.

3D Systems Inc. (2020). *STL File Format.* [viewed, November 2020]. URL: http://www.3dsystems.com/quickparts/learning-center/what-is-stl-file.

3MF Consortium (2020). *3D Manufacturing Format.* [viewed, December 2020]. URL: http://3mf.io/specification/.

Adzhiev V., Cartwright R., Fausett E., Ossipov A., Pasko A., and Savchenko V. (1999). *HyperFun Project: A Framework for Collaborative Multidimensional F-rep Modeling.* Implicit Surfaces 99 Workshop (Bordeaux, France).

Adzhiev V., Kartasheva E., Kunii T., Pasko A., and Schmitt B. (2002). "Cellular-Functional Modeling of Heterogeneous Objects". In: *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*. SMA '02. ACM, pp. 192–203.

Adzhiev V., Kazakov M., Pasko A., and Savchenko V. (2000). "Hybrid System Architecture for Volume Modeling". In: *Computers & Graphics* 24.1, pp. 67–78.

Alexa M., Cohen-Or D., and Levin D. (2000). "As-rigid-as-possible Shape Interpolation". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., pp. 157–164.

Anitescu C., Hossain Md N., and Rabczuk T. (2018). "Recovery-based error estimation and adaptivity using high-order splines over hierarchical T-meshes". In: *Computer Methods in Applied Mechanics and Engineering* 328, pp. 638–662.

Auzinger T., Heidrich W., and Bickel B. (2018). "Computational Design of Nanostructural Color for Additive Manufacturing". In: *ACM Trans. Graph.* 37.4, No. 159, pp. 1–16.

Averbuch-Elor H., Cohen-Or D., and Kopf J. (2016). "Smooth Image Sequences for Data-driven Morphing". In: *Computer Graphics Forum* 35, pp. 203–213.

Bader C., Kolb D., Weaver J. C., and Oxman N. (2016). "Data-Driven Material Modeling with Functional Advection for 3D Printing of Materially Heterogeneous Objects". In: *3D Printing and Additive Manufacturing* 3.2, pp. 71–79.

Bader C., Kolb D., Weaver J. C., Sharma S., Hosny A., Costa J., and Oxman N. (2018). "Making Data Matter: Voxel Printing for the Digital Fabrication of Data Across Scales and Domains". In: *Science Advances* 4.5, No. eaas8652, pp. 1–12.

Bálint C., Valasek G., and Gergó L. (2019). "Operations on Signed Distance Functions". In: Conference: The 11th Conference of PhD Students in Computer Science, pp. 1–12.

Barbier A., Galin E., and Akkouche S. (2005). "A framework for modeling, animating, and morphing textured implicit models". In: *Graphical Models* 67.3, pp. 166–188.

Barclay J., Dhokia V., and Nassehi A. (2016). "Additive Manufacturing Simulation Using Signed Distance Fields". In: *Sustainable Design and Manufacturing 2016.* Ed. by Setchi R., Howlett R. J., Liu Y., and Theobald P. Springer International Publishing, pp. 435–444.

Bastos T. and Celes W. (2008). "GPU-accelerated Adaptively Sampled Distance Fields". In: *2008 IEEE International Conference on Shape Modeling and Applications*, pp. 171–178.

Beier T. and Neely S. (1992). "Feature-based Image Metamorphosis". In: *SIGGRAPH Computer Graphics* 26.2, pp. 35–42.

Belyaevm A., Fayolle P.-A., and Pasko A. (2013). "Signed Lp-distance Fields". In: *Computer-Aided Design* 45.2. Solid and Physical Modeling 2012, pp. 523–528.

Bhashyam S., Shin K. H., and Dutta D. (2000). "An Integrated CAD System for Design of Heterogeneous Objects". In: *Rapid Prototyping Journal* 6.2, pp. 119–135.

Biswas A. and Shapiro V. (2004). "Approximate Distance Fields with Non-Vanishing Gradients". In: *Graphical Models* 66.3, pp. 133–159.

Biswas A., Shapiro V., and Tsukanov I. (2004). "Heterogeneous Material Modeling with Distance Fields". In: *Computer Aided Geometric Design* 21.3, pp. 215–242.

Blinn J. F. (1982). "A Generalization of Algebraic Surface Drawing". In: *ACM Transactions on Graphics* 1.3, pp. 235–256.

Bloomenthal J. and B. Wyvill, eds. (1997). *Introduction to Implicit Surfaces.* Morgan Kaufmann Publishers Inc.

Borgefors G. (1984). "Distance Transformations in Arbitrary Dimensions". In: *Computer Vision, Graphics, and Image Processing* 27.3, pp. 321–345.

Borgefors G. (1986). "Distance Transformations in Digital Images". In: *Computer Vision, Graphics, and Image Processing* 34.3, pp. 344–371.

Borgefors G. (1996). "On Digital Distance Transforms in Three Dimensions". In: *Computer Vision and Image Understanding* 64.3, pp. 368–376.

Botsch M., Pauly M., Kobbelt L., Alliez P., Lévy B., Bischoff S., and Rössl C. (2007). "Geometric Modeling Based on Polygonal Meshes". In: *ACM SIGGRAPH 2007 Courses.* SIGGRAPH '07 1. ACM.

Breen D. E. and Whitaker R. T. (2001). "A level-set approach for the metamorphosis of solid models". In: *IEEE Transactions on Visualization and Computer Graphics* 7.2, pp. 173–192.

Brun E., Guittet A., and Gibou F. (2012). "A local level-set method using a hash table data structure". In: *Journal of Computational Physics* 231.6, pp. 2528–2536.

Brunton A., Arikan C. A., Tanksale T. M., and Urban P. (2018). "3D Printing Spatially Varying Color and Translucency". In: *ACM Transactions on Graphics* 37.4, No. 157, pp. 1–13.

Buhmann M. D. (2003). *Radial Basis Functions: Theory and Implementations.* Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.

Burtnyk N. and Wein M. (1976). "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation". In: *Communications of the ACM* 19.10, pp. 564–569.

Campen M. and Kobbelt L. (2010). "Exact and Robust (Self-)Intersections for Polygonal Meshes". In: *Computer Graphics Forum* 29.2, pp. 397–406.

Careil V., Billeter M., and Eisemann E. (2020). “Interactively Modifying Compressed Sparse Voxel Representations”. In: *Computer Graphics Forum* 39.2, pp. 111–119.

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. Association for Computing Machinery, pp. 67–76.

Cecil T. C., Osher S. J., and Qian J. (2006). “Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension”. In: *Journal of Computational Physics* 213.2, pp. 458–473.

Chandru V., Manohar S., and Prakash C. E. (1995). “Voxel-based modeling for layered manufacturing”. In: *IEEE Computer Graphics and Applications* 15.6, pp. 42–47.

Chen Y., Georgiou T., and Tannenbaum A. (2018). “Vector-Valued Optimal Mass Transport”. In: *SIAM Journal on Applied Mathematics* 78.3, pp. 1682–1696.

Cimquest Inc. and Hewlett-Packard Development Compahy L.P. (2014). *HP Multi Jet Fusion technology.* Tecnical White Paper, pp. 1–8. URL: https://cimquest-inc.com/hp/.

Coifman R. R. and Lafon S. (2006). “Diffusion Maps”. In: *Applied and Computational Harmonic Analysis* 21.1. Special Issue: Diffusion Maps and Wavelets, pp. 5–30.

Corker-Marin Q., Pasko A., and Adzhiev V. (2018). “4D Cubism: Modeling, Animation, and Fabrication of Artistic Shapes”. In: *IEEE Computer Graphics and Applications* 38.03, pp. 131–139.

Crane K., Weischedel C., and Wardetzky M. (2013). “Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow”. In: *ACM Transactions on Graphics* 32.5, No. 152, pp. 1–11.

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.* I3D ’09. Association for Computing Machinery, pp. 15–22.

Cuturi M. and Peyré G. (2016). “A Smoothed Dual Approach for Variational Wasserstein Problems”. In: *SIAM Journal on Imaging Sciences* 9.1, pp. 320–343.

Dado B., Kol T. R., Bauszat P., Thiery J.-M., and Eisemann E. (2016). “Geometry and Attribute Compression for Voxel Scenes”. In: *Computer Graphics Forum* 35.2, pp. 397–407.

Dalstein B., Ronfard R., and Panne M. van de (2015). “Vector Graphics Animation with Time-varying Topology”. In: *ACM Transactions on Graphics* 34.4, No. 145, pp. 1–12.

Danielsson P.-E. (1980). “Euclidean Distance Mapping”. In: *Computer Graphics and Image Processing* 14.3, pp. 227–248.

Darbon J. and Osher S. (2016). “Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere”. In: *Research in the Mathematical Sciences* 3.19.

Delfour M. and Zolsio J.-P. (2010). *Shapes and Geometries: Metrics, Analysis, Differential Calculus, and Optimization.* Society for Industrial and Applied Mathematics.

Deng J., Chen F., Li X., Hu C., Tong W., Yang Z., and Feng Y. (2008). “Polynomial Splines over Hierarchical T-meshes”. In: *Graphical Models* 70.4, pp. 76–86.

Dinh H. Q., Yezzi A., Turk G., and Turk G. (2005). “Texture Transfer During Shape Transformation”. In: *ACM Transactions on Computer Graphics* 24.2, pp. 289–310.

Do Q. H., Mita S., and Yoneda K. (2014). “Narrow passage path planning using fast marching method and support vector machine”. In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, pp. 630–635.

Dolonius D., Sintorn E., Kämpe V., and Assarsson U. (2019). “Compressing Color Data for Voxelized Surface Geometry”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.2, pp. 1270–1282.

Doubrovski E. L., Tsai E. Y., Dikovsky D., Geraedts J. M. P., Herr H., and Oxman N. (2015). “Voxel-based Fabrication Through Material Property Mapping: A Design Method for Bitmap Printing”. In: *Computer-Aided Design* 60. Material Ecology, pp. 3–13.

Drebin R. A., Carpenter L., and Hanrahan P. (1988). “Volume Rendering”. In: *SIGGRAPH Computer Graphics* 22.4, pp. 65–74.

Du H. (2003). “Interactive Shape Design Using Volumetric Implicit PDEs”. In: *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*. SM ’03. Seattle, Washington, USA: ACM, pp. 235–246.

Du H. and Qin H. (2001). “Integrating Physics-Based Modeling with PDE Solids for Geometric Design”. In: *Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001*, pp. 198–207.

Ďurikovič R., Czanner S., and Inoue H. (2001). “Growth animation of human organs”. In: *The Journal of Visualization and Computer Animation* 12.5, pp. 287–295.

Eisemann M., De Decker B., Magnor M., Bekaert P., De Aguiar E., Ahmed N., Theobalt C., and Sellent A. (2008). “Floating Textures”. In: *Computer Graphics Forum* 27.2, pp. 409–418.

Farenick D. (2016). *Fundamentals of Functional Analysis*. Universitext. Springer International Publishing.

Farin G. (2002). *Curves and Surfaces for CAGD*. A volume in The Morgan Kaufmann Series in Computer Graphics.

Farin G. E. (1996). *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Code*. 4th. Academic Press, Inc.

Fayolle P.-A. and Pasko A. (2016). “An evolutionary approach to the extraction of object construction trees from 3D point clouds”. In: *Computer-Aided Design* 74, pp. 1–17.

Fayolle P.-A., Pasko A., and Schmitt B. (2008). “Heterogeneous Objects Modelling and Applications”. In: ed. by Pasko A., Adzhiev V., and Comninos P. Springer-Verlag. Chap. SARDF: Signed Approximate Real Distance Functions in Heterogeneous Objects Modeling, pp. 118–141.

Fernando R. (2004). *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Pearson Higher Education.

Forcadel N., Guyader Le, and Gout C. (2008). “Generalized fast marching method: applications to image segmentation”. In: *Numerical Algorithms* 48, pp. 189–211.

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: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’00. ACM Press/Addison-Wesley Publishing Co., pp. 249–254.

Fryazinov O., Pasko A., and Adzhiev V. (2011). “BSP-fields: An exact representation of polygonal objects by differentiable scalar fields based on binary space partitioning”. In: *Computer-Aided Design* 43.3, pp. 265–277.

Fryazinov O., Sanchez M., and Pasko A. (2015). “Shape Conforming Volumetric Interpolation with Interior Distances”. In: *Computers & Graphics* 46.C, pp. 149–155.

Fryazinov O., Vilbrandt T., and Pasko A. (2013). “Multi-scale Space-variant FRep Cellular Structures”. In: *Computer-Aided Design* 45.1. Computer-aided multi-scale materials and product design, pp. 26–34.

Fu Z., Jeong W.-K., Pan Y., Kirby R. M., and Whitaker R. T. (2011). “A Fast Iterative Method for Solving the Eikonal Equation on Triangulated Surfaces”. In: *SIAM Journal on Scientific Computing* 33.5, pp. 2468–2488.

Gao F., Wu W., Lin Y., and Shen S. (2018). “Online Safe Trajectory Generation for Quadrotors Using Fast Marching Method and Bernstein Basis Polynomial”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 344–351.

Gao L., Chen S.-Y., Lai Y.-K., and Xia S. (2017). “Data-Driven Shape Interpolation and Morphing Editing”. In: *Computer Graphics Forum* 36.8, pp. 19–31.

Giblin P. (2010). *Graphs, Surfaces and Homology*. 3rd ed. Cambridge University Press.

Goes F. de, Goldenstein S., and Velho L. (2008). “A Hierarchical Segmentation of Articulated Bodies”. In: *Proceedings of the Symposium on Geometry Processing*. SGP ’08. Eurographics Association, pp. 1349–1356.

Gómez J. V., Álvarez D., Garrido S., and Moreno L. (2019). “Fast Methods for Eikonal Equations: An Experimental Survey”. In: *IEEE Access* 7, pp. 39005–39029.

Goodman J. E., J. O’Rourke, and C. D. Tóth, eds. (2017). *Handbook of Discrete and Computational Geometry*. Chapman and Hall/CRC.

Guittet A., Lepilliez M., Tanguy S., and Gibou F. (2015). “Solving elliptic problems with discontinuities on irregular domains – the Voronoi Interface Method”. In: *Journal of Computational Physics* 298, pp. 747–765.

Gupta V., Kasana K. S., and Tandon P. (2010). “Computer Aided Design Modeling for Heterogeneous Objects”. In: *CoRR* 1004.3571. arXiv: 1004.3571.

Guthe M., Balázs A., and Klein R. (2005). “GPU-based Trimming and Tessellation of NURBS and T-Spline Surfaces”. In: *ACM Transactions on Graphics* 24.3, pp. 1016–1023.

H. Dyer R. and E. Edmunds D. (2014). *From Real to Complex Analysis*. Springer International Publishing.

Haker S., Zhu L., Tannenbaum A., and Angenent S. (2004). “Optimal Mass Transport for Registration and Warping”. In: *International Journal of Computer Vision* 60.3, pp. 225–240.

Hart J. C. (1996). “Sphere Tracing: a Geometric Method for the Antialiased Ray Tracing of Implicit surfaces”. In: *The Visual Computer* 12.10, pp. 527–545.

Heckbert P. S. and Garland M. (1997). *Survey of Polygonal Surface Simplification Algorithms*. Technical Report, SIGGRAPH’97.

Hiller J. D. and Lipson H. (2009). “Stl 2.0: A Proposal for a Universal Multi-material Additive Manufacturing File Format”. In: pp. 266–278.

Hoetzlein R. K. (2016). “GVDB: Raytracing Sparse Voxel Database Structures on the GPU”. In: *Proceedings of High Performance Graphics*. HPG ’16. Eurographics Association, pp. 109–117.

Hong S. and Jeong W. (2017). “A Group-Ordered Fast Iterative Method for Eikonal Equations”. In: *IEEE Transactions on Parallel and Distributed Systems* 28.2, pp. 318–331.

Hormann K. (2014). “Barycentric Interpolation”. In: *Approximation Theory XIV: San Antonio 2013*. Ed. by Fasshauer G. E. and Schumaker L. L. Springer International Publishing, pp. 197–218.

Hormann K. and Floater M. S. (2006). “Mean Value Coordinates for Arbitrary Planar Polygons”. In: *ACM Trans. Graph.* 25.4, pp. 1424–1441.

Houghton E. G., Emnett R. F., Factor J. D., and Sabharwal C. L. (1985). “Implementation of a Divide-and-Conquer Method for Intersection of Parametric Surfaces”. In: *Computer Aided Geometric Design* 2.1, pp. 173–183.

Hughes J. F., McGuire M., Sklar D. F., Foley J. D., Feiner S. K., and Akeley K. (2014). *Computer Graphics Principles and Practice*. Addison-Wesley.

Igouchkine O., Zhang Y., and Ma K. (2018). “Multi-Material Volume Rendering with a Physically-Based Surface Reflection Model”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.12, pp. 3147–3159.

Intel Corp. (2020). *Open Source Computer Vision Library*. [viewed, November 2020]. URL: http://www.opencv.org/.

Jacob B. and Guennebaud G. (2020). *Open Multi-Processing API*. [viewed, November 2020]. URL: http://www.eigen.tuxfamily.org/.

Jacobson A., Panozzo D., et al. (2018). *libigl: A Simple C++ Geometry Processing Library*. URL: http://www.libigl.github.io/.

Jeong W.-K. and Whitaker R. T. (2008). “A Fast Iterative Method for Eikonal Equations”. In: *SIAM Journal on Scientific Computing* 30.5, pp. 2512–2534.

Jin B. and Geng W. (2015). “Correspondence Specification Learned from Master Frames for Automatic Inbetweening”. In: *Multimedia Tools and Applications* 74.13, pp. 4873–4889.

Jones M. W., Baerentzen J. A., and Sramek M. (2006). “3D Distance Fields: A Survey of Techniques and Applications”. In: *IEEE Transactions on Visualization and Computer Graphics* 12.4, pp. 581–599.

Ju T., Schaefer S., and Warren J. (2005). “Mean Value Coordinates for Closed Triangular Meshes”. In: *ACM Trans. Graph.* 24.3, pp. 561–566.

Kämpe V., Sintorn E., and Assarsson U. (2013). “High Resolution Sparse Voxel DAGs”. In: *ACM Trans. Graph.* 32.4, No. 101, pp. 1–13.

Kantorovich L. V. (1948). “On a Problem of Monge”. In: *Uspekhi Mat. Nauk* 3.2, pp. 225–226.

Karczmarczuk J. (1999). “Geometric Modelling in Functional Style”. In: *Proc. of the III Latino-American Workshop on Functional Programming, CLAPF’99*, pp. 1–14.

Kartasheva E., Adzhiev V., Comninos P., and Fryazinov O. (2008). *Heterogeneous Objects Modelling and Applications*. Ed. by Pasko A., Adzhiev V., and Comninos P. Vol. 4889. Information Systems and Applications, incl. Internet/Web, and HCI. Springer-Verlag Berlin Heidelberg, pp. 2–41.

Kaufman A. E. (1994). “Voxels as a Computational Representation of Geometry”. In: *The Computational Representation of Geometry. SIGGRAPH ’94 Course Notes*.

Keinert B., Schäfer H., Korndörfer J., Ganse U., and Stamminger M. (2014). “Enhanced Sphere Tracing”. In: *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. Ed. by Giachetti A. The Eurographics Association.

Kiciak P. (2016). *Geometric Continuity of Curves and Surfaces.* Synthesis Lectures on Visual Computing. Morgan & Claypool Publishers.

Kim J.-H., Kim C.-H., and Lee J. (2015). “A hybrid SDF for the detailed representation of liquid–solid mixed surfaces”. In: *Computer Animation and Virtual Worlds* 26.5, pp. 527–536.

Kim L., Sukhatme G. S., and Desbrun M. (2004). “A haptic-rendering technique based on hybrid surface representation”. In: *IEEE Computer Graphics and Applications* 24.2, pp. 66–75.

Kiss G., Giannelli C., and Jüttler B. (2014). “Algorithms and Data Structures for Truncated Hierarchical B–splines”. In: *Mathematical Methods for Curves and Surfaces.* Ed. by Floater M., Lyche T., Mazure M.-L., Mørken K., and Schumaker L. L. Springer Berlin Heidelberg, pp. 304–323.

Knott G. D. (2000). *Interpolating Cubic Splines.* Vol. 18. Progress in Computer Science and Applied Logic. Birkhäuser Basel.

Koschier D., Deul C., Brand M., and Bender J. (2017). “An hp-adaptive Discretization Algorithm for Signed Distance Field Generation”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.10, pp. 2208–2221.

Kou X. Y. and Tan S. T. (2007). “Heterogeneous Object Modeling: A Review”. In: *Computer-Aided Design* 39.4, pp. 284–301.

Kou X. Y. and Tan S. T. (2010). “A Simple and Effective Geometric Representation for Irregular Porous Structure Modeling”. In: *Computer-Aided Design* 42.10, pp. 930–941.

Kou X. Y., Tan S. T., and Sze W. S. (2006). “Modeling complex heterogeneous objects with non-manifold heterogeneous cells”. In: *Computer-Aided Design* 38.5, pp. 457–474.

Kumar V., Burns D., Dutta D., and Hoffmann C. (1999). “A Framework for Object Modeling”. In: *Computer-Aided Design* 31.9, pp. 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. ACM, pp. 336–345.

Kumar V. and Dutta D. (1998). “An Approach to Modeling & Representation of Heterogeneous Objects”. In: *Journal of Mechanical Design* 120.4, pp. 659–667.

Lacroute P. and Levoy M. (1994). “Fast Volume Rendering Using a Shear-warp Factorization of the Viewing Transformation”. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH ’94. ACM, pp. 451–458.

Lai M.-J. (2009). “Multivariate Splines and Their Applications”. In: *Encyclopedia of Complexity and Systems Science.* Ed. by Meyers R. A. Springer New York, pp. 5800–5841.

Laine S. and Karras T. (2010). “Efficient Sparse Voxel Octrees”. In: *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.* I3D ’10. Association for Computing Machinery, pp. 55–63.

Lee B., Darbon J., Osher S., and Kang M. (2017). “Revisiting the redistancing problem using the Hopf–Lax formula”. In: *Journal of Computational Physics* 330, pp. 268–281.

Lee S., Wolberg G., and Shin S. Y. (1998). “Polymorph: Morphing Among Multiple Images”. In: *IEEE Computer Graphics and Applications* 18, pp. 58–71.

Lei S., Frank M. C., Anderson D. D., and Brown T. D. (2014). “A Method to Represent Heterogeneous Materials for Rapid Prototyping: the Matryoshka Approach”. In: *Rapid Prototyping Journal* 20.5, pp. 390–402.

Lelièvre P. G., Farquharson C. G., and Hurich C. A. (2011). “Computing first-arrival seismic traveltimes on unstructured 3-D tetrahedral grids using the Fast Marching Method”. In: *Geophysical Journal International* 184.2, pp. 885–896.

Lévy B. (2015). “A numerical algorithm for L2 semi-discrete optimal transport in 3D”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 49.6, pp. 1693–1715.

Leymarie F. and Levine M. D. (1992). “Fast Raster Scan Distance Propagation on the Discrete Rectangular Lattice”. In: *CVGIP: Image Understanding* 55.1, pp. 84–94.

Li B., Fu J., Shang C., and Lin Z. (2020). “Review of heterogeneous material objects modelling in additive manufacturing”. In: *Visual Computing for Industry, Biomedicine, and Art* 3.6, pp. 1–18.

Li Q., Wills D., Phillips R., Viant W. J., Griffiths J. G., and Ward J. (2004). “Implicit Fitting Using Radial Basis Functions with Ellipsoid Constraint”. In: *Computer Graphics Forum* 23.1, pp. 55–69.

Liao J., Lima R. S., Nehab D., Hoppe H., Sander P. V., and Yu J. (2014). “Automating Image Morphing Using Structural Similarity on a Halfway Domain”. In: *ACM Transactions on Graphics* 33.5, No. 168, pp. 1–12.

Lin Y., Chen C., Song M., and Liu Z. (2009). “Dual-RBF Based Surface Reconstruction”. In: *Visual Computer* 25.5-7, pp. 599–607.

Lipman Y., Rustamov R. M., and Funkhouser T. A. (2010). “Biharmonic Distance”. In: *ACM Transactions on Graphics* 29.3, No. 27, pp. 1–11.

Liu Y.-S., Ramani K., and Liu M. (2011). “Computing the Inner Distances of Volumetric Models for Articulated Shape Description with a Visibility Graph”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.12, pp. 2538–2544.

Livesu M., Ellero S., Martínez J., Lefebvre S., and Attene M. (2017). “From 3D Models to 3D Prints: An Overview of the Processing Pipeline”. In: *Computer Graphics Forum* 36.2, pp. 537–654.

Lorensen W. E. and Cline H. E. (1987). “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”. In: *SIGGRAPH Computer Graphics* 21.4, pp. 163–169.

Love A. E. H (2013). *A Treatise on the Mathematical Theory of Elasticity.* 4th Edition. Cambridge University Press.

Ludwig M., Berrier S., Tetzlaff M., and Meyer G. (2015). “3D shape and texture morphing using 2D projection and reconstruction”. In: *Computers & Graphics* 51. International Conference Shape Modeling International, pp. 146–156.

Maas J., Rumpf M., and Simon S. (2017). “Transport Based Image Morphing with Intensity Modulation”. In: *Scale Space and Variational Methods in Computer Vision.* Ed. by Lauze F., Dong Y., and Dahl A. B. Springer International Publishing, pp. 563–577.

Makihara Y. and Yagi Y. (2011). “Earth Mover’s Morphing: Topology-Free Shape Morphing Using Cluster-Based EMD Flows”. In: *Computer Vision – ACCV 2010.* Ed. by Kimmel R., Klette R., and Sugimoto A. Springer Berlin Heidelberg, pp. 202–215.

Markworth A. J., Ramesh K. S., and Parks W. P. (1995). “Modelling Studies Applied to Functionally Graded Materials”. In: *Journal of Materials Science* 30.9, pp. 2183–2193.

Massarwi F., Machchhar J., Antolin P., and Elber G. (2018). “Hierarchical, Random and Bifurcation Tiling with Heterogeneity in Micro-structures Construction via

Functional Composition". In: *Computer-Aided Design* 102. Special Issue on SPM 2018, pp. 148–159.

McCormack J. and Sherstyuk A. (1998). "Creating and Rendering Convolution Surfaces". In: *Computer Graphics Forum* 17.2, pp. 113–120.

Mehlhorn K. (1984). *Data Structures and Algorithms 3: Multi-Dimensional Searching and Computational Geometry.* Springer-Verlag.

Michalatos P. and Payne A. (2018). "Digital porosity". In: *International Journal of Rapid Manufacturing* 7. Issue 2/3, pp. 219–239.

Min C. and Gibou F. (2007). "A second order accurate level set method on non-graded adaptive cartesian grids". In: *Journal of Computational Physics* 225.1, pp. 300–321.

Mirzadeh M., Guittet A., Burstedde C., and Gibou F. (2016). "Parallel level-set methods on adaptive tree-based grids". In: *Journal of Computational Physics* 322, pp. 345–364.

Moore R. E., Kearfott R. B., and Cloud M. J. (2009). *Introduction to Interval Analysis.* Society for Industrial and Applied Mathematics.

Mousavizadeh L. and Ghasemi S. (2020). "Genotype and phenotype of COVID-19: Their roles in pathogenesis". In: *Journal of Microbiology, Immunology and Infection.* DOI: 10.1016/j.jmii.2020.03.022.

Mullikin J. C. (1992). "The Vector Distance Transform in Two and Three Dimensions". In: *CVGIP: Graphical Models and Image Processing* 54.6, pp. 526–535.

Museth K. (2013). "VDB: High-resolution Sparse Volumes with Dynamic Topology". In: *ACM Transactions on Graphics* 32.3, No. 27, pp. 1–22.

Muuss J. M. and Butler A. L. (1991). "Combinatorial Solid Geometry, Boundary Representations, and Non-Manifold Geometry". In: *Advanced Computer Graphics Techniques*, pp. 185–223.

Nader G. and Guennebaud G. (2018). "Instant Transport Maps on 2D Grids". In: *ACM Transactions on Graphics* 37.6, No. 249, pp. 1–13.

Nealen A., Müller M., Keiser R., Boxerman E., and Carlson M. (2006). "Physically Based Deformable Models in Computer Graphics". In: *Computer Graphics Forum* 25.4, pp. 809–836.

Neumann A., Alexander B., and Neumann F. (2017). "Evolutionary Image Transition Using Random Walks". In: *Computational Intelligence in Music, Sound, Art and Design.* Ed. by Correia J., Ciesielski V., and Liapis A. Springer International Publishing, pp. 230–245.

Ngo T. D., Kashani A., Imbalzano G., Nguyen K. T. Q., and Hui D. (2018). "Additive Manufacturing (3D printing): A Review of Materials, Methods, Applications and Challenges". In: *Composites Part B: Engineering* 143, pp. 172–196.

Niino M., Suzuki A., Hirai T., Watanabe R., Hirano T., and N. Kuroishi (1987). "Method of Producing a Functionally Gradient Material". 30. Patent number: US4751099A.

Ohtake Y., Belyaev A., and Seidel H.-P. (2006). "Sparse surface reconstruction with adaptive partition of unity and radial basis functions". In: *Graphical Models* 68.1. Special Issue on SMI 2004, pp. 15–24.

OpenMP Architecture Review Board Corp. (2020). *Open Multi-Processing API.* [viewed, November 2020]. URL: http://www.openmp.org/.

Osher S. and Sethian J. A. (1988). "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations". In: *Journal of Computational Physics* 79.1, pp. 12–49.

Pan R. and Skala V. (2007). "Implicit Surface Modeling Suitable for Inside/Outside Tests with Radial Basis Functions". In: *2007 10th IEEE International Conference on Computer-Aided Design and Computer Graphics*, pp. 28–28.

Park S.-M., Crawford R. H., and Beaman J. J. (2000). "Functionally Gradient Material Representation by Volumetric Multi-Texturing for Solid Freeform Fabrication". In: *Proceedings of the 11th Solid Freeform Fabrication Symposium*, pp. 360–361.

Pasko A. and Adzhiev V. (2004). "Function-Based Shape Modeling: Mathematical Framework and Specialized Language". In: *Automated Deduction in Geometry*. Ed. by Winkler F. Springer Berlin Heidelberg, pp. 132–160.

Pasko A., Adzhiev V., Schmitt B., and Schlick C. (2001). "Constructive Hypervolume Modeling". In: *Graphical Models* 63.6, pp. 413–442.

Pasko A., Adzhiev V., Sourin A., and Savchenko V. (1995). "Function Representation in Geometric Modeling: Concepts, Implementation and Applications". In: *The Visual Computer* 11.8, pp. 429–446.

Pasko A., Fryazinov O., Vilbrandt T., Fayolle P.-A., and Adzhiev V. (2011). "Procedural Function-based Modelling of Volumetric Microstructures". In: *Graphical Models* 73.5, pp. 165–181.

Pasko G. I., Pasko A. A., and Kunii T. L. (2005). *Bounded blending for function-based shape modeling.*

Patané G. (2016). "STAR: Laplacian Spectral Kernels and Distances for Geometry Processing and Shape Analysis". In: *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics: State of the Art Reports.* EG '16. Eurographics Association, pp. 599–624.

Patanè G. and Spagnuolo M. (2012). "SMI 2012: Full Local Approximation of Scalar Functions on 3D Shapes and Volumetric Data". In: *Computer Graphics* 36.5, pp. 387–397.

Payne B. A. and Toga A. W. (1992). "Distance Field Manipulation of Surface Models". In: *IEEE Computer Graphics and Applications* 12.1, pp. 65–71.

Perlin K. and Hoffert E. M. (1989). "Hypertexture". In: *SIGGRAPH Comput. Graph.* 23.3, pp. 253–262.

Perry R. N. and Frisken S. F. (2001). "Kizamu: A System for Sculpting Digital Characters". In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH '01. Association for Computing Machinery, pp. 47–56.

Piegl L. and Tiller W. (1997). *The NURBS Book*. Springer-Verlag.

Plegl L. and Tiller W. (1995). *The NURBS Book.* Monographs in Visual Communication. Springer-Verlag Berlin Heidelberg.

Qian X. and Dutta D. (2004). "Feature-based Design for Heterogeneous Objects". In: *Computer-Aided Design* 36, pp. 1263–1278.

Qin Y., Qi Q., Scott P. J., and Jiang X. (2019). "Status, comparison, and future of the representations of additive manufacturing data". In: *Computer-Aided Design* 111, pp. 44–64.

Regli W., Rossignac J., Shapiro V., and Srinivasan V. (2016). "The New Frontiers in Computational Modeling of Material Structures". In: *Computer-Aided Design* 77.C, pp. 73–85.

Rehman T. U., Pryor G., and Tannenbaum A. (2007). "Fast Multigrid Optimal Mass Transport for Image Registration and Morphing". In: *Proceedings of the British Machine Vision Conference.* Ed. by Rajpoot N. M. and Bhalerao A. H. BMVA Press, No. 11, pp. 1–10.

Reiner T., Mückl G., and Dachsbacher C. (2011). “Interactive modeling of implicit surfaces using a direct visualization approach with signed distance functions”. In: *Computers & Graphics* 35.3. Shape Modeling International (SMI) Conference 2011, pp. 596–603.

Rosenfeld A. and Pfaltz J. L. (1966). “Sequential Operations in Digital Picture Processing”. In: *Journal ACM* 13.4, pp. 471–494.

Rossignac J. R. and Requicha A. A. G. (1986). “Offsetting Operations in Solid Modelling”. In: *Computer-Aided Geometric Design* 3.2, pp. 129–148.

Royston M., Pradhana A., Lee B., Chow Y. T., Yin W., Teran J., and Osher S. (2018). “Parallel redistancing using the Hopf–Lax formula”. In: *Journal of Computational Physics* 365, pp. 7–17.

Rubner Y., Tomasi C., and Guibas L. J. (2000). “The Earth Mover’s Distance as a Metric for Image Retrieval”. In: *International Journal of Computer Vision* 40.2, pp. 99–121.

Rustamov R. M. (2011). “Interpolated Eigenfunctions for Volumetric Shape Processing”. In: *The Visual Computer* 27, No. 951, pp. 1–11.

Rustamov R. M., Lipman Y., and Funkhouser T. (2009). “Interior Distance Using Barycentric Coordinates”. In: *Proceedings of the Symposium on Geometry Processing*. SGP ’09. Eurographics Association, pp. 1279–1288.

Rvachev V. L. (1973). *Methods of Logic Algebra in Mathematical Physics.* (rus). Kiev, Ukraine: Naukova Dumka.

Rvachev V. L. (1982). *Theory of R-functions and Some Applications.* (rus). Naukova Dumka.

Rvachev V. L., Sheiko T. I., Shapiro V., and Tsukanov I. (2001). “Transfinite interpolation over implicitly defined sets”. In: *Computer-Aided Geometric Design* 18.3, pp. 195–220.

Samozino M., Alexa M., Alliez P., and Yvinec M. (2006). “Reconstruction with Voronoi Centered Radial Basis Functions”. In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. SGP ’06. Eurographics Association, pp. 51–60.

Sanchez M., Fryazinov O., Adzhiev V., Comninos P., and Pasko A. (2015). “Space-Time Transfinite Interpolation of Volumetric Material Properties”. In: *IEEE Transactions on Visualization and Computer Graphics* 21.2, pp. 278–288.

Sanchez M., Fryazinov O., Fayolle P.-A., and Pasko A. (2015). “Convolution Filtering of Continuous Signed Distance Fields for Polygonal Meshes”. In: *Computer Graphics Forum* 34.6, pp. 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. Association for Computing Machinery, pp. 101–108.

Satherley R. and Jones M. W. (2001). “Vector-City Vector Distance Transform”. In: *Computer Vision and Image Understanding* 82.3, pp. 238–254.

Satyan L. D. and O’Rourke J. (2011). *Discrete and Computational Geometry*. Princeton University Press. URL: http://press.princeton.edu/titles/9489.html.

Schechter E. (1997). “Chapter 25 - Fréchet Derivatives”. In: *Handbook of Analysis and Its Foundations*. Ed. by Schechter E. Academic Press, pp. 661–687.

Schmidt R., Wyvill B., Sousa M. C., and Jorge J. A. (2007). “ShapeShop: Sketch-Based Solid Modeling with BlobTrees”. In: *ACM SIGGRAPH 2007 Courses*. SIGGRAPH ’07. Association for Computing Machinery, pp. 1–43.

Schmitz G. J. (2016). “Microstructure Modeling in Integrated Computational Materials Engineering (ICME) Settings: Can HDF5 Provide the Basis for an Emerging Standard for Describing Microstructures?” In: *JOM* 68.1, pp. 77–83.

Schmitz G. J., Böttger B., Apel M., Eiken J., Laschet G., Altenfeld R., Berger R., Boussinot G., and Viardin A. (2016). “Towards a Metadata Scheme for the Description of Materials – the Description of Microstructures”. In: vol. 17. 1. Taylor & Francis, pp. 410–430.

Schumacher C., Bickel B., Rys J., Marschner S., Daraio C., and Gross M. (2015). “Microstructures to Control Elasticity in 3D Printing”. In: *ACM Transactions on Graphics* 34.4, No. 136, pp. 1–13.

Schumaker L. (2007). *Spline Functions: Basic Theory*. 3rd ed. Cambridge Mathematical Library. Cambridge University Press.

Serra J. P. (1988). *Image Analysis and Mathematical Morphology*. Academic Press.

Shapiro V. (1999). “Well-Formed Set Representations of Solids”. In: *International Journal of Computational Geometry & Applications* 09.02, pp. 125–150.

Shapiro V. and Tsukanov I. (1999). “Implicit Functions with Guaranteed Differential Properties”. In: *Proceedings of the Fifth ACM Symposium on Solid Modeling and Applications*. SMA ’99. ACM, pp. 258–269.

Sharma G. K. and Gurumoorthy B. (2017). “A Hybrid Approach to Define and Represent Material Distribution in Heterogeneous Objects”. In: *Computer-Aided Design and Applications* 14.1, pp. 70–82.

Shu C.-W. and Osher S. (1989). “Efficient implementation of essentially non-oscillatory shock-capturing schemes, II”. In: *Journal of Computational Physics* 83.1, pp. 32–78.

Side Effects Software Inc. (2020). *SideFX Houdini*. [viewed, November 2020]. URL: http://www.sidefx.com/.

Silicon Graphics Inc. (2020). *Open Graphics Library*. [viewed, November 2020]. URL: http://www.opengl.org/.

Simar T. (2020). *Magicavoxel Software*. [viewed, October 2020]. URL: http://www.voxelmade.com/magicavoxel/.

Siu Y. K. and Tan S. T. (2002a). “‘Source-based’ Heterogeneous Solid Modeling”. In: *Computer-Aided Design* 34.1, pp. 41–55.

Siu Y. K. and Tan S. T. (2002b). “Modeling the Material Grading and Structures of Heterogeneous Objects for Layered Manufacturing”. In: *Computer-Aided Design* 34.10. Rapid Technologies: Solutions for Today and Tomorrow, pp. 705–716.

Smelik R. M., Tutenel T., Bidarra R., and Benes B. (2014). “A Survey on Procedural Modelling for Virtual Worlds”. In: *Computer Graphics Forum* 33.6, pp. 31–50.

Smith A. R. (1987). “Planar 2-pass Texture Mapping and Warping”. In: *SIGGRAPH Computer Graphics* 21.4, pp. 263–272.

Solomon J., Goes F. de, Peyré G., Cuturi M., Butscher A., Nguyen A., Du T., and Guibas L. (2015). “Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains”. In: *ACM Transactions on Graphics* 34.4, No. 66, pp. 1–11.

Solomon J., Rustamov R., Guibas L., and Butscher A. (2014). “Earth Mover’s Distances on Discrete Surfaces”. In: *ACM Trans. Graph.* 33.4, No. 67, pp. 1–12.

Strain J. (2000). “A Fast Modular Semi-Lagrangian Method for Moving Interfaces”. In: *Journal of Computational Physics* 161.2, pp. 512–536.

Stratasys Ltd. (2014). *Objet500 Connex3*. white paper, p. 7. URL: https://www.stratasys.com/.

Subcommittee F42.04 (2020). *Specification for additive manufacturing file format (AMF) Version 1.2 (ISO / ASTM52915-20)*. URL: http://www.astm.org/cgi-bin/resolver.cgi?ISOASTM52915.

Sullivan A. (2015). *Hybrid adaptively sampled distance fields*. Patent number: 9122270, United States, US patent.

Sussman M., Smereka P., and Osher S. (1994). "A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow". In: *Journal of Computational Physics* 114.1, pp. 146–159.

Svensson S. and Borgefors G. (2002). "Digital Distance Transforms in 3D Images Using Information from Neighbourhoods up to 5×5×5". In: *Computer Vision and Image Understanding* 88.1, pp. 24–53.

Takahashi T., Masumori A., Fujii M., and Tanaka H. (2018). *FAV (Fabricatable Voxel) File Format Specification*. Tech. rep. Fuji Xerox Co. Ltd. & Social Fabrication Laboratory of Keio University at Shonan-Fujisawa Campus (SFC).

Tam C. K. W. and Kurbatskii K. A. (2000). "A Wavenumber Based Extrapolation and Interpolation Method for Use in Conjunction with High-Order Finite Difference Schemes". In: *Journal of Computational Physics* 157.2, pp. 588–617.

Tang Y. and Feng J. (2018). "Multi-scale surface reconstruction based on a curvature-adaptive signed distance field". In: *Computers & Graphics* 70. CAD/Graphics 2017, pp. 28–38.

Tereshin A., Adzhiev V., Fryazinov O., Marrington-Reeve F., and Pasko A. (2020). "Automatically Controlled Morphing of 2D Shapes with Textures". In: *SIAM Journal on Imaging Sciences* 13.1, pp. 78–107. DOI: doi.org/10.1137/19M1241581.

Tereshin A., Adzhiev V., Fryazinov O., and Pasko A. (2019). "Hybrid Function Representation with Distance Properties". In: *Eurographics 2019 - Short Papers*. Ed. by Cignoni P. and Miguel E. The Eurographics Association, pp. 17–20. DOI: doi.org/10.2312/egs.20191004.

Tereshin A., Anderson E., Pasko A., and Adzhiev V. (2020). "Space-Time Blending for Heterogeneous Objects". In: *Eurographics 2020 - Short Papers*. Ed. by Wilkie A. and Banterle F. The Eurographics Association, pp. 45–48. DOI: doi.org/10.2312/egs.20201014.

Tereshin A., Pasko A., Fryazinov O., and Adzhiev V. (2020). *Hybrid Function Representation for Heterogeneous Objects*. arXiv: 2012.15176 [cs.GR]. URL: https://arxiv.org/abs/2012.15176.

Tereshin A., Pasko A., Fryazinov O., and Adzhiev V. (2021). "Hybrid function representation for heterogeneous objects". In: *Graphical Models* 114, p. 101098. DOI: doi.org/10.1016/j.gmod.2021.101098.

The GNU Triangulated Surface Library (2020). [viewed, November 2020]. URL: http://www.gts.sourceforge.net/.

Tsitsiklis J. N. (1995). "Efficient algorithms for globally optimal trajectories". In: *IEEE Transactions on Automatic Control* 40.9, pp. 1528–1538.

Tsukanov I. and Posireddy S. R. (2011). "Hybrid Method of Engineering Analysis: Combining Meshfree Method with Distance Fields and Collocation Technique". In: *Journal of Computing and Information Science in Engineering* 11, No. 031001, pp. 1–9.

Turk G. and O'Brien J. F. (2002). "Modelling with Implicit Surfaces that Interpolate". In: *ACM Transactions on Graphics* 21.4, pp. 855–873.

Versprille K. J. (1975). "Computer-Aided Design Applications of the Rational b-Spline Approximation Form." AAI7607690. PhD thesis.

W. Olver F., Lozier D., Boisvert R., and Clark C. (2010). *NIST Handbook of Mathematical Functions*. Cambridge University Press.

Wang J., Yang Z., Jin L., Deng J., and Chen F. (2011). “Parallel and Adaptive Surface Reconstruction Based on Implicit PHT-splines”. In: *Computer Aided Geometric Design* 28.8. Solid and Physical Modeling 2010, pp. 463–474.

Wang Y. (1996). “Intersection of Offsets of Parametric Surfaces”. In: *Computer Aided Geometric Design* 13.5, pp. 453–465.

Weng Y., Chai M., Xu W., Tong Y., and Zhou K. (2013). “As-Rigid-As Possible Distance Field Metamorphosis”. In: *Computer Graphics Forum* 32.7, pp. 381–389.

Westover L. (1990). “Footprint Evaluation for Volume Rendering”. In: *SIGGRAPH Computer Graphics* 24.4, pp. 367–376.

Westover L. A. (1991). “Splatting: A Parallel, Feed-forward Volume Rendering Algorithm”. UMI Order No. GAX92-08005. PhD thesis.

Wolberg G. (1998). “Image morphing: a survey”. In: *The Visual Computer* 14.8, pp. 360–372.

Wu C., Deng J., and Chen F. (2007). “Fast Data Extrapolating”. In: *Journal of Computational and Applied Mathematics* 206.1, pp. 146–157.

Wu Z. and Sullivan Jr J. M. (2003). “Multiple Material Marching Cubes Algorithm”. In: *International Journal for Numerical Methods in Engineering* 58.2, pp. 189–207.

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* 18, pp. 149–158.

Wyvill G., Mcpheeters C., and Wyvill B. (1986). “Data Structure for Soft Objects”. In: *Visual Computer* 2, pp. 227–234.

Yang N., Quan Z., Zhang D., and Tian Y. (2014). “Multi-morphology Transition Hybridization CAD Design of Minimal Surface Porous Structures for Use in Tissue Engineering”. In: *Computer-Aided Design* 56, pp. 11–21.

Ye Q.-Z. (1988). “The signed Euclidean Distance Transform and Its Applications”. In: *9th International Conference on Pattern Recognition (IEEE Cat. No.88CH2614-6)*. Vol. 1. IEEE Computer Society, pp. 495–499.

Yin K., Liu Y., and Wu E. (2011). “Fast Computing Adaptively Sampled Distance Field on GPU”. In: *Pacific Graphics Short Papers*. Ed. by Chen B.-Y., Kautz J., Lee T.-Y., and Lin M. C. The Eurographics Association.

Yngve G. and Turk G. (2002). “Robust Creation of Implicit Surfaces from Polygonal Meshes”. In: *IEEE Transactions on Visualization and Computer Graphics* 8.4, pp. 346–359.

Yongbin J., Liguan W., Lin B., and Jianhong C. (2009). “Boolean Operations on Polygonal Meshes Using OBB Trees”. In: *2009 International Conference on Environmental Science and Information Application Technology*. Vol. 1, pp. 619–622.

Yoo D. (2012). “Heterogeneous Minimal Surface Porous Scaffold Design Using the Distance Field and Radial Basis Functions”. In: *Medical Engineering & Physics* 34.5, pp. 625–639.

Yuan Z., Yu Y., and Wang W. (2012). “Object-space Multiphase Implicit Functions”. In: *ACM Transactions on Graphics* 31.4, No. 114, pp. 1–10.

Zanni C., Claux F., and Lefebvre S. (2018). “HCSG: Hashing for Real-Time CSG Modeling”. In: *Proceedings of the ACM in Computer Graphics Interactive Techniques* 1.1, No. 15, pp. 1–19.

Zhao H. (2004). “A Fast Sweeping Method for Eikonal Equations”. In: *Mathematics of Computation* 74.250, pp. 603–627.

Zhu L., Yang Y., Haker S., and Tannenbaum A. (2007). “An Image Morphing Technique Based on Optimal Mass Preserving Mapping”. In: *Transactions on Image Processing* 16.6, pp. 1481–1495.