CONSTRUCTION OF VOLUMETRIC OBJECT MODELS
USING DISTANCE-BASED SCALAR FIELDS

By
Pierre-Alain Fayolle

A DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN COMPUTER SCIENCE AND ENGINEERING

Graduate Department of Information Systems
The University of Aizu
2006
The thesis titled

*Construction of Volumetric Object Models using Distance-Based Scalar Fields*

by

Pierre-Alain Fayolle

is reviewed and approved by:

Main referee

*Professor*

Nikolay Mirenkov


*Professor*

Gennadiy Nikishkov


*Professor*

Minetada Osano


Hosei University

*Professor*

Alexander Pasko


The University of Aizu

2006
# Table of Contents

Table of Contents

List of Figures

List of Tables

Abstract

1 Introduction

1.1 Objectives ................................................. 1

1.2 Structure of the dissertation ................................ 3

1.3 List of external publications ................................. 4

2 Previous works

2.1 Representation of solids ..................................... 7

2.1.1 Enumerative ........................................ 7

2.1.2 Implicit ........................................... 8

2.1.3 Function Representation ................................. 9

2.1.4 Boundary Representation ............................... 11

2.2 Construction and approximation of the signed Euclidean distance function ........................................ 12

2.2.1 The signed Euclidean distance function ................ 12

2.2.2 Computation of the distance function ................. 13

2.2.3 Constructive geometry with distance functions ........ 14

2.3 Heterogeneous material modeling .......................... 21

2.3.1 Techniques for heterogeneous material modeling .... 21

2.3.2 Discussion .......................................... 24

2.4 Automation of modeling and reverse engineering ........ 24

2.4.1 Creation of triangle meshes and implicit surface fitting .... 25
3 Theoretical approaches

3.1 SARDF operations
   3.1.1 A first approach: circular min approximation
   3.1.2 Circular min approximation with bounded radius
   3.1.3 SARDF intersection construction
   3.1.4 Smoothness of the SARDF intersection
   3.1.5 SARDF framework

3.2 Automation of modeling by template FRep models
   3.2.1 Template FRep models
   3.2.2 Fitting problem formulation
   3.2.3 Nonlinear minimization of least squares by local methods
   3.2.4 Simulated annealing
   3.2.5 Genetic algorithms
   3.2.6 Discussion

3.3 Constructive tree recovery using a genetic algorithm
   3.3.1 Description of the algorithm
   3.3.2 Relation with genetic programming
   3.3.3 Description of solids and primitives

4 Implementation and details of the algorithms and methods

4.1 The HyperFun project
   4.1.1 HyperFun language
   4.1.2 HyperFun software tools

4.2 Web-based shape modeling using real distance functions
   4.2.1 HyperFun applet components
   4.2.2 Extension for distance-based modeling
   4.2.3 Examples of distance-based modeling with the applet

4.3 HyperFun models with approximate distance-based functions
   4.3.1 The HyperFun interpreter
   4.3.2 SARDF extension

4.4 Fitting template FRep model
   4.4.1 Implementation of an hybrid method
   4.4.2 Using a genetic algorithm for the global search
   4.4.3 FRep evaluation for the fitting algorithm

4.5 Implementation’s details of the constructive tree recovery algorithm
4.5.1 The genotype, and its decoding (finding an object’s phenotype from its genotype) ........................................ 102
4.5.2 The genetic operations ........................................ 104

5 Experimental results and applications 106
5.1 Constructive modeling with SARDF: quality of approximation .... 106
  5.1.1 Comparison of different implementations for intersection .... 106
  5.1.2 Comparison of the gradients’ norms for different implementations of the intersection ......................... 108
  5.1.3 Qualitative comparison of the union operations: SARDF union, max, and R-union ................................. 114
  5.1.4 Comparison of SARDF operations, min/max and R-functions with the exact distance function .................. 116
5.2 Constructive heterogeneous objects modeling with SARDF .......... 120
  5.2.1 Two-dimensional example ................................... 120
  5.2.2 Three-dimensional CAD part .................................. 126
5.3 Constructive modeling with SARDF: applications ................... 129
  5.3.1 Metamorphosis of functionally defined shapes .............. 129
  5.3.2 Constant radius offsetting .................................... 133
  5.3.3 Tapered extrusion ............................................. 134
5.4 Experiments with fitting template FRep models ....................... 138
  5.4.1 First CAD part .................................................. 138
  5.4.2 Lacquer ware sake pot ....................................... 140
  5.4.3 Fitting template models made with SARDF ................. 144
  5.4.4 Application in Finite Element Meshes (FEM) ............... 146
5.5 Constructive tree recovery using a genetic algorithm ............... 149
  5.5.1 Example 1 ..................................................... 149
  5.5.2 Example 2 ..................................................... 151
  5.5.3 Comments ..................................................... 152

6 Conclusion 153
6.1 Main contributions ............................................... 153
6.2 Future works and possible extensions ................................ 156

Acknowledgments 157

Bibliography 158
List of Figures

2.1 Example of a constructive tree with primitives in the leaves and operations in the internal nodes. The intermediate shapes are illustrated in the internal nodes. The final object is at the bottom. 10

2.2 Some contour lines of: the "distance" function to a unit square defined by the intersection, using min, of four infinite lines (left), and the exact signed distance function to the boundary of the square (right). See the circular arc in the exterior contour line (green) of the exact distance function (right), compared to the sharp corners created by using min (left). 16

2.3 Illustration of the $C^1$ discontinuity of min/max in further operations, in this case: the blending union between a sphere and a cube. 17

2.4 Illustration of a nice blending effect, when R-functions are used as set-theoretic operations during modeling. 19

2.5 Left, "distance" to a circle of radius 5 obtained as the union of the disk with itself, the maximum of the function reaches 17.071; right, distance to a circle of radius 5, the maximum value is 5, reached at the center. 20

2.6 Left, A simple T like shape made by the union of two blocks. Right, the set-theoretic operations are replaced by their blending counterpart. 30

3.1 Each contour line of the min function contains a sharp corner. 35
3.2 Contour line of the function min has a sharp corner (left) to be replaced by a circular arc (right). ........................................... 36
3.3 Two straight lines with the angle $\theta$ between them break the first quadrant into three zones. .................................................. 36
3.4 New contour line configuration: the two rays are attached to the circular arc at the junction points $P_1$ and $P_2$. .................................. 37
3.5 Unknowns of the Eq. 3.1.1 and their geometric relations. ............... 38
3.6 The continuation of $L_1$ and $L_2$ from the quadrant 1 ($x > 0$ and $y > 0$) breaks the quadrant 3 ($x < 0$ and $y < 0$ into three new zones $D$, $E$, and $F$. ................................................................. 40
3.7 Unknowns of the Eq. 3.1.3 and their geometric relations. ............... 41
3.8 The radius of the circular arc is growing with the distance from the origin, thus increasing the error in the distance function made by the approximation. ......................................................... 43
3.9 Contour lines of the circular min (left) and max (right) approximations; note the use of different threshold radius. ......................... 43
3.10 The first quadrant is divided into two zones. The circular approxima-
tion is applied in zone I, whereas we introduce a fixed radius approxi-
mation with the bounding band in zone II. ............................ 44
3.11 Geometric relations between the parameters of the circular approxima-
tion with the bounding band in zone II. ............................ 46
3.12 The third quadrant is divided into two zones. The circular approxi-
mation is applied in zone III, whereas we introduce a fixed radius approxi-
mation with the bounding band in zone IV. ............................ 48
3.13 Geometric relations between the parameters of the circular approxima-
tion with the bounding band in zone IV. ............................ 50
3.14 Illustration of the general idea for the construction of SARDF intersection. The two possible approaches in the quadrant 3 are given left and right. Left, the growth of the radius is bounded by two straight lines; right, the growth of the radius is unbounded. 

3.15 The different zones of the SARDF intersection in quadrant 1

3.16 The different zones in the second possible approach in the quadrant 3

3.17 Illustration of a parameterized template FRep model: a model is built with FRep in a constructive way with abstract parameters that can be tuned to satisfy some modeling criteria.

3.18 Illustration of a crossover operation acting on two parents and producing two offsprings.

3.19 Illustration of a mutation operation altering one gene.

3.20 Illustration of a segmented point-set with primitives fitted to the subsets. Each primitive is associated to one color.

3.21 Additional halfspaces may be required to build a constructive representation from a boundary representation. The left object is a boundary representation of a simple two-dimensional solid, where each boundary patch has a different color. The right object contains an additional halfspace, which is needed to build a constructive representation of the solid.

4.1 System architecture for the HyperFun project from [1]

4.2 Example of an HyperFun model

4.3 User interface of the HyperFun applet for distance-based shape modeling.

4.4 Menu for the selection of the set-theoretic operations.

4.5 Blending operations: (a) Initial object composed as a vertical cone, a torus-like shape and a block shape. (b) Blending between the block and the cone. (c) Blending between the torus and the cone. (d) Blending between the block and the cone, and blending between the previous resulting object with the torus.
4.6 Screenshot of the menu for generating cross-sections of the model. . . 87
4.7 Polygonal model of a complex artistic vase modeled using the Java
HyperFun applet and distance functions and rendered within the applet
(Model by Benjamin Schmitt). . . . . . . . . . . . . . . . . . . . . . . 88
4.8 High quality photo-realistic rendering of the vase modeled using the
distance-based HyperFun applet, and rendered using the POV-Ray
raytracer. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90
4.9 Raytraced mechanical CAD object modeled using our applet and dis-
tance functions. The original STL file for the CAD object is courtesy
of Elena Kartasheva. . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
4.10 Distance field of the CAD object at the cross-section by the plane
z = 0. SARDF operations are defined with a radius of 0.5. . . . . . . . 92
4.11 Example of an HyperFun model using SARDF operations, R-functions
and distance-based primitives . . . . . . . . . . . . . . . . . . . . . 95

5.1 Contour plots of different implementations of intersection: min (top
left), R-function (top right), SARDF intersection (bottom) . . . . . . . 107
5.2 Contour plot of the gradient’s norm of the R-function \( (x, y) \rightarrow x + y - \sqrt{x^2 + y^2} \). The values of the norm range from 0.414214 to 2.41421. . . 110
5.3 Contour plots of the gradient’s norm of the SARDF intersection, when
the bounded radius approach (first approach) is used in the third quad-
rant. Values of the radius are respectively: top left: 0.1, 0.1; top right:
0.5, 0.5; bottom: 5.0, 5.0 . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
5.4 Contour plots of the gradient’s norm of the SARDF intersection, when
the angle approach (second approach) is used in the third quadrant.
Values of the radius (first quadrant) and the angle (third quadrant)
are respectively: top left: 0.1, 0.0; top right: 0.1, 0.2; bottom: 0.1, 0.5 112
5.5 Contour maps of the union of two ellipsoids for the three different union operations. The level of gray corresponds to ranges of distances from points inside the solid to its surface. Left: SARDF union, middle: max: sharp corners indicate a $C^1$ discontinuity, right: R-union: contour map of the function clearly shows that the resulting function loses the distance like behaviour quite close to the boundary even with exact distance functions used for the arguments.  

5.6 Two-dimensional CAD part. (a) Geometry of the CAD part defined by an FRep model. (b) Three different material regions (blue: material 1, red: material 2, color gradient: functionally graded material).  

5.7 Approximate distance map $d_1$ from point $X$ to the boundary of the region where only material 1 exists. (a) Using R-functions. (b) Using SARDF operations.  

5.8 Approximate distance map $d_2$ from point $X$ to the boundary of the region where only material 2 exists. (a) Using R-functions. (b) Using SARDF operations.  

5.9 A cross-section parallel to $x$-axis and the distribution of the materials in the cross section for the CAD part constructed with SARDF functions.  

5.10 Material distributions in the cross-section $y = 2$ for materials 1 and 2 using: left, min/max in the constructive trees for the geometry of the solid and the material regions. The circled points correspond to points of $C^1$ discontinuity of the material distributions. Right: R-functions in the constructive trees for the geometry of the solid and the material regions.  

5.11 Material distribution for the CAD part: the zone in blue contains only the material 1, the zone in red contains only the material 2, the zone in gradient corresponds to a region where both materials are blended, the weight of each material is given by Eq. 5.2.1 and 5.2.2.
5.12 (a) the first material feature, (b) the second material feature, with a zoom on one of the pins, on the right, (c) union of the two material features (Model made by Benjamin Schmitt).

5.13 Distribution of two materials. Blue color corresponds to material 1, red color to material 2. The color variation indicates the fraction of each material. (a) Two cross sections are made for \( x = 0 \) and \( y = 0 \) to show the material distribution. (b) A zoom is made on one of the pins with two additional cross-sections.

5.14 Left, a Buddha shape, initial shape for the metamorphosis; right, the final shape for the metamorphosis: a kanji, meaning Buddha.

5.15 Intermediate frames of the metamorphosis between the Buddha shape, and the kanji with R-functions used as set-theoretic operations during modeling.

5.16 Intermediate frames of the metamorphosis between the Buddha shape, and the kanji with SARDF functions used to implement set-theoretic operations during modeling.

5.17 Constant radius offsetting for an L shape consisting in concave and convex polygons.

5.18 Tapered extrusion of a rectangle: top view (left) and the cross-section A (right).

5.19 Tapered extrusion of a convex generator (top, left) modeled with R-functions (top, right), min/max (bottom, left), and SARDF operations (bottom, right).

5.20 Tapered extrusion of the concave generator with the hole: generator (top left), contour map of its defining function (top right), and tapered extrusion (bottom).

5.21 Data set used for the reverse engineering of the mechanical part. It contains 10714 3D points, scattered on the surface of the part.
5.22 Shapes for the best fitted FRep in two cases: (right) the best fitted object does not correspond to the real object when starting with set2 and using the QN method; (left) the best fitted object corresponds to the real object when starting with the set2 and using the hybrid method.

5.23 Evolution of the shape during the fitting process using the hybrid method.

5.24 Local minimum effect: result of the fitting with a local method starting with wrong initial parameters. The point-set for the sake pot is also displayed for comparison.

5.25 Evolution of the shapes of the sake pot during the fitting process.

5.26 The reconstructed object.

5.27 A surface mesh, generated by a CAD system (top, left), the recovered shape (top, right), the associated optimized mesh (bottom, left), and the 3D tetrahedral mesh generated from it (bottom, right). The original BRep model and the generated optimized and 3D tetrahedral meshes are courtesy of Elena Kartasheva.

5.28 Evolution of the fitness of the best individual for a population of size 1000.

5.29 Result for the best individual for the first mechanical part.

5.30 Evolution of the fitness of the best individual for a population of size 1000.

5.31 Result for the best individual for the second mechanical part.
## List of Tables

5.1 Value of the norm of the SARDF gradient for different values of the parameters, when the first approach in the third quadrant for constructing the SARDF intersection is used.  

5.2 Value of the norm of the SARDF gradient for different values of the parameters, when the second approach in the third quadrant for constructing the SARDF intersection is used.  

5.3 Min, max, mean and variance of the error between approximate distance functions created using SARDF operations, min/max, or R-functions and the distance function.  

5.4 The error from the distance function to each approximate distance using the error metric Eq. (5.1.4).  

5.5 Two sets of initial values used for fitting: set1 is close to the best set of values, in contrary to set2.  

5.6 Time (in seconds) taken by each method to converge to the best fit and least-square error (sum of the deviations squared) of the best fit for each set of initial values. QN stands for Quasi-Newton, SA for simulated annealing and SAQN is an hybrid algorithm consisting in a combination of simulated annealing and Quasi-Newton.  

5.7 Initial and final vectors of parameters obtained by fitting the sake pot model with a Quasi-Newton method.
5.8 Final vector of parameters after using the hybrid approach: a global minimization method (simulated annealing) followed by a local minimization method (Newton) .............................................. 143

5.9 Best fitted parameters and squared error for a template model made by: (1) R-functions and algebraic primitives and (2) R-functions and distance primitives ................................................................. 145

5.10 Best fitted parameters and squared error for a template model made by: (3) Min/max and distance primitives and (4) SARDF operations and distance primitives ......................................................... 145

5.11 Fitting function value for the best fit set of parameters and the corresponding mean error ................................................................. 147
Abstract

We consider in this dissertation the modeling of volumetric objects by distance-based scalar fields. The Euclidean distance to a set, which is defined by the shortest distance to any points of the set, is a concise yet powerful way to define volumetric objects or solids; it also has a physical and intuitive meaning useful for many shape and solid modeling applications. Consider a closed curve or surface in Euclidean space, the signed distance to the boundary defines a solid – the interior correspond to the set of points at a positive distance, the exterior to the set of points at a negative distance. Starting from expressions of the Euclidean distance to primitives (plane, sphere, torus), it is possible to construct more complex shapes by applying set-theoretic operations (union, intersection, difference) to them. In the theory of R-function [82], due to Rvachev, these set-theoretic operations are expressed by real valued functions; applying them to primitives, which are also real valued functions, gives for the final function defining the complex shape a functional expression. This general formulation was extended by Pasko et al. [70] to form the theory of Function Representation (FRep), but uses mostly algebraic distances for the primitives.

It is well-known that the R-functions destroy quickly the Euclidean distance approximation for the resulting function, whereas min/max (another functional expression for set operations) keep a better approximation but add points of $C^1$ discontinuity for the resulting function. We consider in this dissertation functional expressions for the set-theoretic operations that keep a reasonable approximation of the Euclidean distance but also provide smoothness for the resulting function, a property useful for
many applications. We call these functions SARDF for Signed Approximate Real Distance Functions. The framework made by these set operations and primitives defining the Euclidean distance to closed curves or surfaces is introduced in this dissertation. This framework is implemented within the HyperFun Java applet and the HyperFun interpreter and used for heterogeneous material modeling, where the distance to the shape boundary and material features is used to parameterize the material distribution.

Modeling objects in a constructive way, i.e. by recursively applying set-theoretic operations to primitives is a well-known and powerful paradigm in solid modeling. Combined with the functional expression of the final solid and the Euclidean distance property, it provides a powerful tool for solid modeling and applications. The construction of objects following this constructive paradigm may however be tedious and sometimes repetitive. Automation of the process is suitable and for that purpose we introduce the notion of template constructive FRep. In order to reuse existing FRep models, we propose to parameterize them and to fit the parameters to different point-sets for optimizing and adapting the shape to different models either obtained from laser scanning or from existing meshes.

Fitting of non-linear parameterized FRep models is made by a combination of meta-heuristics such as simulated annealing or genetic algorithms with local methods such as Newton methods.

Automation of the creation of a constructive FRep is also considered by creating constructive FRep from a segmented point-set and a list of primitives. A genetic algorithm is used to find the best constructive expression involving the primitives fitted to the segmented point-set and operations selected from a set of possible operations. The obtained constructive FRep can be parameterized and used as a valid template FRep, reused and fitted to different instances of the object.

The dissertation is organized as follow:

**Chapter 1, Introduction:** In this chapter, we present the objectives of the thesis; give the structure of the dissertation and the principal contributions. These
contributions are:

- New functions and their properties for the implementation of the set-theoretic operations (intersection, union, difference), which are smooth and keep a good approximation of the Euclidean distance function.
- Introduction of a new framework combining the previous functions with primitives defined by the signed Euclidean distance function.
- Use of the above framework for modeling constructive heterogeneous objects.
- Introduction of the concept of parameterized template FRep and presentation of new algorithms for fitting the parameterized template FRep to point-sets.
- Introduction of a new algorithm for creating constructive FRep from a segmented point-set and the associated fitted primitives.

**Chapter 2, Previous works:** Related work is discussed in Chapter 2. We describe in more details the different available models to define solids and volumetric objects with emphasis on the Function Representation model. The Euclidean distance function is a subset of the Function Representation and one of the possible concise ways to define volumetric objects. A survey of the different possible constructions of the Euclidean distance function and its approximation is given. The strengths and weaknesses of the existing methods are discussed. We then consider the problem of modeling automation in the case of existing shapes acquired by scanning (whether it is laser scanning, MRI scanning, CT or other methods) or available in a different format (BRep, mesh) and the associated problems such as: segmentation of point-set, fitting, and others.

**Chapter 3, Theoretical approaches:** This chapter presents a solution to the problem of modeling objects defined by the Euclidean distance function in a constructive way by introducing a new formulation for the set-theoretic operations which is smooth and keeps a good approximation of the distance function. We explain how to combine these set-theoretic operations with primitives defined by the
signed Euclidean distance function in a general framework named SARDF for Signed Approximate Real Distance Function. Some techniques to generate distance fields to freeform complex primitives are given to extend the framework. Modeling shapes in a constructive way can be a difficult and time consuming task, in order to automate this process, we introduce the notion of parameterized FRep models. Parameterized family of similar shapes is discussed as well as its fitting to different scanned point-sets as a way to automate the modeling process. Finally, automating the creation of parameterized models is discussed with a new algorithm to recover constructive trees from segmented point-sets.

**Chapter 4, Implementation and details of the algorithms and methods:** Here we consider the software implementations of the proposed algorithms and methods. Some of these implementations are done within the HyperF un environment and extend it. HyperF un is a language and a set of tools for shape and solid modeling. An extension of the HyperF un applet is presented for supporting approximate distance modeling. The implementations’ details of the algorithms for fitting template parameterized FRep and recovering constructive tree of an FRep model are given.

**Chapter 5, Experimental results and applications:** This chapter describes some experimental results and some applications of the methods and algorithms described in this dissertation. The framework for approximate constructive distance modeling is used first to model mechanical homogeneous solids and heterogeneous solids. We also illustrate the use of constructive modeling with distance function in shape metamorphosis, constant radius offsetting and tapered extrusion. Template parameterized modeling and fitting is illustrated through the fitting of a sake pot from the Virtual Shikki project, and the fitting of different mechanical parts. Finally, we illustrate the automatic generation of constructive trees from segmented point-sets for real mechanical parts.

**Chapter 6, Conclusion:** We conclude on the results of the proposed methods, their advantages and problems. This leads to some extra needs that will be considered as future work.
Chapter 1

Introduction

1.1 Objectives

Modeling objects, their properties and relations is an important topic in computer science. In this dissertation, objects are geometrical point-sets, and we are interested in their modeling and construction using distance-based scalar fields. Spatial objects do not only refer to curves in two-dimension and surfaces in three-dimension but also to their interior and their internal properties (color, material, physical and others); the use of distance-based scalar fields can naturally define them and serve to describe these internal properties. We use the term volumetric objects to refer to them.

Modeling and visualization of volumetric objects have multiple applications including: cultural heritage preservation, scientific visualization, physical/chemical analysis, mechanical engineering, simulation, and others.

Several mathematical models have been developed to construct volumetric objects; all of them have their own strengths and weaknesses and depend on the field of application. In this dissertation, we consider volumetric objects defined by real continuous distance-based scalar field like for example in the theory of Function Representation [70]. In fact, in the Function Representation (FRep) model, volumetric objects (solids) are defined by a continuous real-valued function of point coordinates. FRep generalizes implicit surfaces, constructive solid geometry, and other shape and solid modeling techniques in a uniform framework.
In this work, we restrict our interest to the constructive modeling paradigm and the creation of shapes and solids defined by the signed Euclidean distance function or its approximation. Constructive modeling is an elegant way to build solids using a tree data structure with operations in the nodes and primitives in the leaves. The tree, called a constructive tree, keeps information about the structure, construction operations and semantics of the objects. The signed Euclidean distance function defines a solid by giving at each point in space the shortest distance between the current point and any point belonging to the surface of the solid. The sign is used to distinguish between interior and exterior.

Distance-based scalar fields present the advantage to naturally define a volume and to simplify the task of modeling. Using the Euclidean distance gives additional advantages such as a physical meaning which makes the modeling task easier by providing a natural parameter to define constraints in the modeling or internal properties and attributes.

The work presented in this dissertation has several objectives:

- The definition of new functional expressions for the set-theoretic operations (intersection, union, and difference), which provide good approximation of the Euclidean distance and smoothness of the resulting function.

- The introduction of a constructive framework – a subset of the FRep model – based on these set-theoretic operations, and other operations and primitives defined by an approximation of the signed Euclidean distance function for distance-based modeling of volumetric objects.

- The extension of the constructive hypervolume model [69] with the above framework to allow parameterization and control of the attributes by the distance.

- The automation of the modeling task, by introducing template parameterized models and recovery of constructive models from segmented point-sets.
1.2 Structure of the dissertation

Related work is discussed in Chapter 2. We describe in more details the different available models to define solids and volumetric objects with emphasis on the Function Representation model. The Euclidean distance function is a subset of the Function Representation and one of the possible concise way to define volumetric objects. A survey of the different possible constructions of the Euclidean distance function and its approximation is given. The strengths and weaknesses of the existing methods are discussed. We then consider the problem of modeling automation in the case of existing shapes acquired by scanning (whether it is laser scanning, MRI scanning, CT or other methods) or available in a different format (BRep, mesh) and the associated problems such as: segmentation of point-set, fitting, and others.

The chapter 3 presents a solution to the problem of modeling objects defined by the Euclidean distance function in a constructive way by introducing new formulations for the set-theoretic operations which are smooth and keep a good approximation of the distance function [29, 28, 31]. We explain how to combine these set-theoretic operations with primitives defined by the signed Euclidean distance function (or an approximation) in a general framework named SARDF for Signed Approximate Real Distance Function. Some techniques to generate distance fields to freeform complex primitives are given to extend the framework. Modeling shapes in a constructive way can be a difficult and time consuming task, in order to automate this process, we introduce the notion of parameterized FRep models. Parameterized family of similar shapes is discussed as well as its fitting to different scanned point-sets as a way to automate the modeling process [25, 30, 26, 108]. Finally, automating the creation of parameterized models is discussed with a new algorithm to recover constructive trees from segmented point-sets [27].

In chapter 4, the software implementations of the proposed algorithms and methods are presented. Some of these implementations are done within the HyperFun environment and extend it. HyperFun is a language and a set of tools for shape and solid modeling [1]. An extension of the HyperFun applet is presented for supporting
approximate distance modeling [31]. The implementations’ details of the algorithms for fitting template parameterized FRep [26] and recovering a constructive tree model are given [27].

The chapter 5 describes some experimental results and some applications of the methods and algorithms described in this dissertation. The framework for approximate constructive distance modeling is used first to model mechanical homogeneous solids and heterogeneous solids [28]. We also illustrate the use of constructive modeling with distance function in shape metamorphosis, constant radius offsetting and tapered extrusion. Template parameterized modeling and fitting is illustrated through the fitting of a sake pot from the Virtual Shikki project [73], and the fitting of different mechanical parts [25]. Finally, we illustrate the automatic generation of constructive tree from segmented point-sets for real mechanical parts.

1.3 List of external publications

The main elements of this dissertation have been published in international journals and proceedings of conference.

- Journals:

- **Proceedings of Conference:**
  
  
  
Chapter 2

Previous works

In this chapter, we briefly review the different models used to represent solids by a computer. A particular attention is given to the Function Representation or FRep model [70], because of the several advantages it provides over the other representations. The Euclidean distance from a point $p$ to a set $S$ is the minimum distance, with the distance defined from the Euclidean norm, between $p$ and any points of $S$. The signed Euclidean distance function is a subset of the FRep model and appears to be a concise and powerful way to represent solids: we present the different methods used to construct Euclidean distance functions to define volumetric objects.

Constructive modeling is a powerful way to create complex solids by combining together simple solids with operations. It can be implemented within the FRep model using the theory of R-functions [82, 84, 91, 70] or min/max [87, 79]. We review the existing methods and algorithms to model in a constructive way volumetric objects with a focus on the quality of approximation of the Euclidean distance and the smoothness of the final resulting function.

Constructive modeling of objects defined by an approximate distance function has various applications in shape and solid modeling. A very interesting one is the construction of solid with heterogeneous material: we review the existing methods of modeling heterogeneous material solids.

Constructing a complex object is a tedious task, and we look for automating that process by the use of parameterized models. We look at the existing methods
for automating the modeling process for existing shapes and solids acquired from scanner devices. The problem of automation modeling includes sub-problems such as: segmenting point-sets, fitting primitives, and the final model construction.

2.1 Representation of solids

First, we briefly describe the different computational models used to traditionally represent solids. Following the classification of [94], there are two main categories:

- enumerative (and combinatorial)
- implicit (and constructive)

2.1.1 Enumerative

Mapping

Enumerative approaches specify rules for generating points belonging to the set. The most popular approach within this category is probably to use a mapping with a parametric definition. Points from a parametric space are mapped to the geometrical space by the mean of functions. B-Spline, Bezier patches or Non Uniform Rational B-Splines (NURBS) belong for example to this category [32]. This representation is popular in Computer Aided Design (CAD) but suffers from important limitations:

- point membership classification requires complex numerical procedures
- difficulty to define operations on these objects, such as the set-theoretic operations.

Groupings or spatial occupancy enumeration

Groupings is considered as the simplest way to represent sets: the enumeration consists of a collection of cells with the same geometric type and dimension. The most popular example from this category is voxels (collection of three dimensional cubes). Octrees [16] and Binary Space Partition (BSP) [102] fall also in this category. The
main problem for this representation is the difficulty to define operations on it: set-theoretic operations or even rigid body motion requires processing and reconstruction.

**Cell complexes**

Cell complexes represent an extension to groupings in the sense that the relationships between cells are included in the model (notions of incidence of cells). The main advantage of the cell complexes is that they represent explicitly all the topological information, so no extra numerical computations are needed to answer topological queries for example. The drawback of this model is the redundancy of information that needs to be encoded, and the need to ensure the validity of the representation at all times.

### 2.1.2 Implicit

Instead of enumerating the points belonging to a set, it is possible to define a geometrical set $S$ implicitly as the set of points satisfying a predicate $P$: $S = \{ p : P(p) = \text{true} \}$. A popular predicate consists in using the sign of a real-valued function $f$: for example the inequality $f(p) \geq 0$ [82].

**Constructive Geometry and Constructive Solid Geometry**

Constructive geometry refers to the construction of complex solids by applying set operations to simple primitives. It can be implemented on a computer by using a tree data structure with set operations in the internal nodes and primitives in the leaves. The point membership classification is simply implemented by a recursive processing of the tree.

Constructive Solid Geometry or CSG [78] is a constructive representation scheme relying on closed regular primitives, rigid body motions and regularized set operations. A primitive defined by a set $S$ is closed regular if: $S = \text{closure}(\text{interior}(S))$. Regularization of set operations implies some additional computation to maintain the neighborhood information for points on primitives and their combinations. It can be a
difficult task. Complete working solutions have been implemented for CSG modeling \cite{14}; simplicity of the data structures and the computation (with recursive algorithms) made it a popular representation in academia and some commercial systems.

From a practical point of view, the methods based on constructive geometry provide naturally an history of the construction, allowing for an editing of the elements, and reflect the logical structure of the object.

An example of a constructive tree is illustrated in Fig. 2.1. The final object made from cylinders and a box is shown at the bottom, and the different steps in the construction of the object are illustrated in the internal nodes of the tree.

A constructive tree (and consequently a CSG tree) can be syntactically translated to a single real-valued functions using the theory of R-functions due to Rvachev \cite{82, 84, 91, 92, 70}. The theory of R-functions has been extended by Pasko et al. into the theory of Function Representation \cite{70}. More details on the theory of Function Representation are given in the following.

\subsection{2.1.3 Function Representation}

Function Representation or FRep is a generalization of implicit surfaces, constructive geometry, sweeping and other shape or solid models \cite{70}. In FRep, a geometric object $M$ is expressed by the sign of a function. The interior of $M$ correspond to $iM = \{p \in E^n : f(p) > 0\}$; the boundary of $M$ correspond to $\partial M = \{p \in E^n : f(p) = 0\}$; and the exterior of $M$ to $eM = \{p \in E^n : f(p) < 0\}$, where $E^n$ is the $n$–dimensional Euclidean space.

Two- and three-dimensional, but also time dependent objects or objects of higher dimension are covered by the FRep model \cite{69}. In general, in the FRep framework an object is represented by a tree data structure (similar to the constructive geometry and CSG) reflecting the logical structure and construction of the object. The leaves of the tree structure contain primitives and the internal nodes arbitrary operations. Primitives can include: algebraic surfaces \cite{13}, convolution surfaces \cite{59}, and others. Various operations have been formulated that keep the FRep representation closed:
Figure 2.1: Example of a constructive tree with primitives in the leaves and operations in the internal nodes. The intermediate shapes are illustrated in the internal nodes. The final object is at the bottom.
applying these operations on FRep results in a valid FRep. Examples of operations are: set-theoretic [82, 87, 79], blending set-theoretic operations [72], offsetting [72], and others.

A possible concise, elegant and powerful way to describe shapes and solids is to use the signed Euclidean distance function. The signed Euclidean distance function is included in the FRep model: if $d$ is the signed Euclidean distance to a boundary, then the interior of the solid can be defined by $\{ p : d(p) > 0 \}$, the boundary by $\{ p : d(p) = 0 \}$ and the unbounded exterior by $\{ p : d(p) < 0 \}$. However, the FRep model is much more general and tends to mix algebraic distance, Euclidean distance and other type of distance functions. A comparison between Euclidean and algebraic distances to quadric surfaces [37] suggests that the Euclidean one is preferable in terms of robustness and accuracy.

### 2.1.4 Boundary Representation

Boundary Representation or BRep can be seen as a compromise between the different approaches – enumerative and implicit – presented before. The solid is defined implicitly by its boundary following the predicate: $S = \{ p : p \in \text{the set bounded by } \partial S \}$ and rely on the fact that $\partial S$ separates the Euclidean space $E^3$ in two subsets: the interior bounded by $\partial S$ and the unbounded exterior. Point membership query on such a set is done by ray casting and counting the number of intersections with the boundary. The boundary can be represented by any way that can guarantee the unambiguous definition of the solid’s boundary and the possibility to compute the intersection with an arbitrary line segment. In order to be valid, the solid’s boundary should respect the following properties:

- be a valid cell complex
- be homogeneously two-dimensional
- every edge is shared by an even number of faces
- be orientable (the normals are globally and consistently oriented)
A natural way to implement the BRep model is to use a combinatorial representation with enumeration of vertices, edges, and faces, and their relations (the topology). Boundary representations can be more compact than the cellular complexes, but it inherits the same default: difficulty to construct, difficulty to maintain after applying operations.

2.2 Construction and approximation of the signed Euclidean distance function

2.2.1 The signed Euclidean distance function

The signed Euclidean distance function\(^1\) from a point \(p \in \mathbb{R}^n\) to a \((n - 1)\) closed orientable manifold \(M\), embedded in \(\mathbb{R}^n\), is defined by: \(d : \mathbb{R}^n \to \mathbb{R}, d(p) = \epsilon|p - c|\), where \(\epsilon\) is \(\pm 1\) corresponding to the orientation of \(M\), \(c\) is the closest point on \(M\) to \(p\), and \(|.|\) denotes the Euclidean norm. Two conventions exist for the sign of the distance: the outward normal can point in the positive direction or in the negative direction. In the case of non-orientable or non-closed \((n - 1)\)-manifolds, the signed distance has no real meaning, but an unsigned distance can be defined by: \(d : \mathbb{R}^n \to \mathbb{R}, d(p) = |p - c|\), using the same notation as above. In this dissertation, we adopt the convention that the outward normal points in the direction of the negative values of the distance function. In the Euclidean three-dimensional space \(\mathbb{R}^3\), the signed Euclidean distance function to a closed oriented surface \(M\) naturally defines a solid by: \(\{(x, y, z) \in \mathbb{R}^3 : d((x, y, z), M) \geq 0\}\).

Euclidean distance fields already have numerous applications in geometric modeling [35], shape metamorphosis [15], object reconstruction from cross-sections [47], robust rendering with sphere tracing [39], generation of skeletal shape representation [114], and other areas.

\(^1\)In the rest of the dissertation we will use the term distance function to refer to the Euclidean distance function
2.2.2 Computation of the distance function

Let \( d(\mathbf{p}), \mathbf{p} \in \mathbb{R}^3 \) be the signed distance function to an oriented closed surface \( M \). The function \( d \) is the viscosity solution of the Eikonal equation [113, 104, 90]:

\[
|\nabla d| = 1, d|_M = 0 \quad (2.2.1)
\]

Physically, \( d \) corresponds to the time arrival of a wave propagating from the surface boundary, with a speed of unit magnitude. Let \( \mathbf{c} \) be the closest point of \( \mathbf{p} \) on the surface \( M \), the distance is then \( |\mathbf{p} - \mathbf{c}| \), with a negative sign if \( \mathbf{p} \) is outside \( M \). If the surface is smooth, then \( \mathbf{p} - \mathbf{c} \) is orthogonal to the surface. The signed Euclidean distance function is at least \( C^0 \), but may be not differentiable at some points.

Expressions for the distance function to most of the classic surfaces of a CSG system (sphere, cylinder, cone) are known analytically [39]. For example, the signed distance to a sphere (boundary of a ball) of radius 1 and center at the point \((0, 0, 0)\) is given by the function: \( d(x, y, z) = 1 - \sqrt{x^2 + y^2 + z^2} \). The signed distance to ellipsoids can be computed by a numerical procedure [40]. The algorithm as given in [40] is however not numerically robust. Suppose that the ellipsoid is centered in \((0, 0, 0)\) with \( e_x, e_y \) and \( e_z \) for its main axis, the algorithm is numerically unstable for all points close to or lying on the planes: \( x = 0, y = 0, \) or \( z = 0 \). This algorithm needs to be extended to treat all these particular cases robustly.

In general, if the surface \( M \) is available as an oriented point-set or a mesh of triangles, it is possible to solve the Eikonal equation 2.2.1 on a finite grid. There exist various good numerical algorithms to solve that problem such as the fast marching method [90], the fast sweeping method [104, 113], or the characteristics / scan conversion algorithm [58]. Algorithms, that exploit the GPU (Graphics Processing Unit), have also been designed in order to compute efficiently the Euclidean distance function [41, 99]. After the signed Euclidean distance has been computed on each nodes of the grid, it is always possible to apply spline interpolation / approximation, to get an analytical expression – see for example the work of Roessl for the interpolation / approximation of volume data [80].
These methods may suffer from numerical issues and a loss of accuracy depending on factors such as the choice of the basis, the sampling of the discrete distance field, or the quality of the input data. Such methods are however attractive for already existing objects acquired, for example, by various scanning devices (like laser scanner, MRI, CT, and others). These methods associated with a further interpolation require that the shape, to which the distance has to be computed, already exists as a mesh or an oriented point-set. It is a rather unpractical scheme for modeling. Indeed, when modeling a shape, it would be required to generate a triangle mesh approximation of it, and then compute the distance function. Modifications of the model, even if a few, would require a recomputation of the distance function. When solids are modeled with a constructive approach which is the case for some solid modeler, and used in some engineering fields, it would appear logical to support building the distance function in a constructive way as well.

Supporting constructive modeling with distance functions implies that the primitives of the system are exact or approximation of the distance function, and that the operations used in the modeling system are keeping the distance property of the resulting function or at least provide an approximation of the distance. Existing formulations for the set-theoretic operations used in constructive modeling are reviewed in the following.

2.2.3 Constructive geometry with distance functions

In constructive geometry, complex solids are built by applying successively set-theoretic operations to primitives. When solids are described by the sign of a function, like in implicit surfaces or FRep, set-theoretic operations can be easily defined. When the primitives have the distance function property – meaning that the value of the function is the signed Euclidean distance to the shape’s boundary – we want that the resulting function for the complex solid, obtained by applying the set-theoretic operations to primitives, is again the distance function or at least a good approximation. We study in the following different implementations of the set operations: min/max
and the R-functions $R_0$ in terms of distance approximation and smoothness.

In the following, $d_1$ and $d_2$ are two distance functions to two $(n - 1)$-manifolds $M_1$ and $M_2$; practically, $n = 2$ or $n = 3$, so $M_1$ and $M_2$ are curves or surfaces, and $d_1$ and $d_2$ naturally define surfaces or solids, denoted by $S_1$ and $S_2$. The results remain valid in any dimension $n$.

**min/max**

Sabin [87] and Ricci [79], independently proposed the use of min/max to simulate set-theoretic operations on implicit surfaces. Using min/max, the set-theoretic operations (union, intersection, difference) are given by:

\[
S_1 \cup S_2 = d_1 \lor d_2 = \max(d_1, d_2) \quad (2.2.2)
\]

\[
S_1 \cap S_2 = d_1 \land d_2 = \min(d_1, d_2) \quad (2.2.3)
\]

\[
S_1 \setminus S_2 = d_1 \land -d_2 \quad (2.2.4)
\]

It is easy to check that the function built using min or max on two distance functions $d_1$ and $d_2$ is 0 on the surface defined by the corresponding set-theoretic operations applied on the solids $S_1$ and $S_2$. If the gradient is defined, its norm is equal to 1; so that both of the properties of Eq. 2.2.1 hold. However, it is not the correct Euclidean distance function. It can be seen in the simple 2D example, Fig. 2.2, with the distance to a square built as an intersection of 4 infinite lines. Some contour lines of the distance function constructed analytically by applying min on infinite lines and the correct distance to the boundary are shown left and right respectively. Exterior contour lines are in green, interior contour lines in blue, and the square shape is in black. The correct distance function has different exterior contour lines in green, with circular arcs centered at the vertices of the square, instead of sharp corners.
Figure 2.2: Some contour lines of: the “distance” function to a unit square defined by
the intersection, using min, of four infinite lines (left), and the exact signed distance
function to the boundary of the square (right). See the circular arc in the exterior
contour line (green) of the exact distance function (right), compared to the sharp
corners created by using min (left).

The major problem with the use of min/max in shape and solid modeling is
related with the smoothness of the functions min and max. \((x, y) \rightarrow \min(x, y)\)
and \((x, y) \rightarrow \max(x, y)\) are \(C^0\) but not differentiable at points where \(x = y\); for
example, in the case of min: \(\min(x, y) = \frac{1}{2}(x + y - |x - y|)\), and \(x \rightarrow |x|\) is not
differentiable in \(x = 0\). In the geometric space, the resulting function will generally
not be differentiable for every points \(p\) such that: \(d_1\) is not differentiable, or \(d_2\) is not
differentiable, or \(d_1(p) = d_2(p)\). The first two cases are inherent to the primitives,
but the latter is added by the min/max functions.

These points can cause unexpected results in further operations on the object such
as blending, metamorphosis, and others, and problems in engineering applications
requiring non-vanishing gradients [9, 10].

Figure 2.3 illustrates the unexpected result of the blending union between a sphere
and a box, when min/max are used in the modeling to implement the set-operations.
The box is defined as an intersection of planes. The blending union is defined by:
\(\text{blending}(d_1, d_2) = d_1 + d_2 + \sqrt{d_1^2 + d_2^2 + \frac{an}{1+(\frac{d_1}{a_1})^2+(\frac{d_2}{a_2})^2}}\)
following the work of Pasko [72].
The unwanted edge appearing in the material added by the blending union comes
from the use of min to implement the intersection in the cube.
Figure 2.3: Illustration of the $C^1$ discontinuity of min/max in further operations, in this case: the blending union between a sphere and a cube.
There are works trying to remove the $C^1$ discontinuities of min/max: Ricci [79] proposed the superelliptic approximations of min/max. These functions do not describe exact set-theoretic operations and suit only for blending. The elliptic approximation of min/max by Barthe et al [3] is designed initially for blending and the error of the distance function grows infinitely far from the boundary. Rvachev proposed the R-functions [82, 83, 91], which are discussed in the following.

**R-functions**

There are various kinds of R-functions, with different order of smoothness, discussed in the following papers [82, 84, 91]. The most commonly used (found in shape and solid modeling literacy) are given by:

\[
S_1 \cup S_2 = d_1 \vee d_2 = d_1 + d_2 + \sqrt{d_1^2 + d_2^2}
\]  
(2.2.5)

\[
S_1 \cap S_2 = d_1 \wedge d_2 = d_1 + d_2 - \sqrt{d_1^2 + d_2^2}
\]  
(2.2.6)

\[
S_1 \setminus S_2 = d_1 \wedge -d_2
\]  
(2.2.7)

The R-functions, $(x, y) \rightarrow x \wedge y$ and $(x, y) \rightarrow x \vee y$, are in $C^1$ over $\mathbb{R}^2 \setminus (0, 0)$. In the geometric space – i.e. when the R-functions are applied to primitives $d_1$ and $d_2$ – the resulting function is not differentiable at all points $p$ such that: $d_1$ is not differentiable, or $d_2$ is not differentiable, or $d_1(p) = d_2(p) = 0$. The first two cases are inherent to the primitives and the latter is added by the R-functions. It corresponds to the sharp corners and sharp edges of a surface. Using R-functions to implement the set-operations, the blending operation does not create unwanted edge as in Fig. 2.3 and produces nice smooth shape as in Fig. 2.4.

If $C^k$ is required, the following R-functions can be used:

\[
S_1 \cup S_2 = d_1 \vee d_2 = d_1 + d_2 + (d_1^k + d_2^k)^{\frac{1}{k}}
\]  
(2.2.8)
Figure 2.4: Illustration of a nice blending effect, when R-functions are used as set-theoretic operations during modeling.
\[ S_1 \cap S_2 = d_1 \land d_2 = d_1 + d_2 - (d_1^k + d_2^k)^{\frac{k}{2}} \]  
(2.2.9)

\[ S_1 \setminus S_2 = d_1 \land -d_2 \]  
(2.2.10)

R-functions generate however a poor approximation of the signed distance function. They suffer from a value growth’s explosion when for example applying them to overlapping solids; figure 2.5 compares the distance to a circle of radius 5 resulting of the union of a disk with itself when the union is defined by an R-function and the distance to the circle alone: the union reaches a value of 17.071 at the center against 5 for the disk alone.

![Figure 2.5](image)

Figure 2.5: Left, ”distance” to a circle of radius 5 obtained as the union of the disk with itself, the maximum of the function reaches 17.071; right, distance to a circle of radius 5, the maximum value is 5, reached at the center.

Discussion

Neither min/max nor the R-functions provide at the same time a reasonable approximation of the distance and smoothness of the resulting function. R-functions are poor approximations of the Euclidean distance, and min/max are even less smooth than the Euclidean distance function (see Fig. 2.2). We should notice however that
smoothness and exact distance are somehow contradictory, since the distance function is by definition not always differentiable (it is not differentiable at all the points belonging to the medial axis of the solid). But we accept to loose some accuracy in favor of smoothness.

We introduce in this dissertation new smooth approximations for min/max operations inspired by the works [3, 44]. The proposed functions are $C^1$ continuous and keep a controllable approximation of the distance function. From this point of view, we call the constructed defining function of the object by the term Signed Approximate Real Distance Function (SARDF), the approximate min function can be called SARDF intersection, and the approximate max function - SARDF union.

These set-theoretic operations form the basis of our modeling system with the primitives defined by signed real distance functions or an approximation. Blending operations can be derived from the set-theoretic ones, and other isometric operations (sometimes also referred as rigid body motions or Euclidean motions) complete them.

2.3 Heterogeneous material modeling

2.3.1 Techniques for heterogeneous material modeling

Solid modeling methods have mostly focused on developing models that capture only the geometry of objects, under the assumption that most of them are homogeneous. Recently, a particular attention has been paid to heterogeneous object modeling. In heterogeneous object modeling, an object has a number of non-uniformly distributed attributes assigned at each point and varying in space. These attributes may or not be continuous and are of different natures such as photometric, material density or distribution, physical, and others. Heterogeneous objects are widely used in different areas such as CAD/CAM/CAE, rapid prototyping, physical simulations, geological and medical modeling.

Several techniques to model heterogeneous objects are available, presenting some noticeable analogies with the representations for homogeneous object modeling presented in Section 2.1.
R-sets are considered as a basis for modeling and are extended for material inclusion in [54]. An object is subdivided in components; each of them is homogeneous inside and has an assigned material index. Set-theoretic operations can be applied to the solid’s components with the corresponding operations on the material. Unfortunately, such modeling techniques are limited to the representation of discretely varying material properties. In [53], a more general model is proposed: the geometry is represented by the point set decomposition into a finite set of closed 3-cells, whereas the attributes are defined by a collection of functions, which map the object geometry to several attributes. Such a mathematical model is known as a fiber bundle, with the geometrical model playing the role of the base space. Several other works are using the same model, extending it in various directions ([8, 21]). However, as noticed in [10], such a model does not really offer concrete computational solutions.

Volumetric representations naturally define solids due to their inherent nature. A homogeneous object can be defined as a subset of the 3D space, with an additional scalar value given at any point. In the case of a spatial enumeration, like voxels [77], extension to heterogeneity consists in adding a scalar value for each attribute [65]. The drawback of this method is the difficulty to directly describe the material distribution, without using a data acquisition device (therefore it is supposed that the object to be modeled already exists). Furthermore, the discrete property of the model requires some special approximation procedures.

A continuous volumetric representation was proposed in [76], where a B-spline volume is used to model the object geometry, whereas the attributes are modeled by means of diffusion. This model seems to suffer from the lack of flexibility of the geometry limited to volume splines.

The constructive hypervolume model is introduced in [69] as a mathematical model to define heterogeneous objects. A general hypervolume object is defined as a multi-dimensional point set \( G \) with multiple attributes given at any of its points. The attributes \( S_i \) represent abstract values or physical characteristics such as temperature,
color, material distribution, etc. A representation of the hypervolume is proposed as:

\[ o = (G, A_1, \ldots, A_k) : (F(X), S_1(X), \ldots, S_k(X)) \]

where:

- \( X = (x_1, \ldots, x_n) \) is a point in \( n \)-dimensional Euclidean space \( E^n \),
- \( F : E^n \rightarrow \mathbb{R} \) is a real-valued defining function of point coordinates to represent point sets \( G \), based on the FRep model [70]
- \( S_i : SP_i \rightarrow \mathbb{R}, SP_i \subset E^n \) is a real-valued scalar function corresponding to an attribute \( A_i \) that is not necessarily continuous.

The function \( F(X) \) is a real valued function. For each given point, the function is evaluated and depending on the sign of the returned value, one can classify the given point as inside, outside or on the boundary of the object. This function is represented in the modeling system by a tree structure with primitives in the leaves and operations in the nodes. The only requirement of the FRep model is that the defining function \( F \) has to be at least \( C^0 \) continuous.

Similarly, depending on the applications, the attribute functions can be defined using physical models or a constructive approach. The spatial subset where an attribute is defined, is called a space partition designated as \( SP_i \) in the above formulation. There are no definite values for an attribute outside its space partition. For each material feature, there is at least one space partition, containing this material feature. However a material feature can be contained in more than one space partition, in the case for example when the material feature is made of the known composition of several materials.

Both the geometry and the space partitions of the object can be defined through constructive modeling, using either the general R-functions [82, 83], or min/max functions [79, 87]. However, the problem of parameterization and control of attributes in the case of material modeling is left unanswered in this model.
Biswas et al. [10] are interested in the representation and control of material distributions by some intuitive parameters related to the geometry of the solid and/or its material features for mesh-free modeling. They propose to use the distance functions from material features (point-sets of any dimension with known material properties) as these parameters. It appears indeed from the existing literature that the (Euclidean) distance, or functions of the distance function are the most common types of material functions constructed by methods based on spatial discretization [98, 46, 55]. The authors of [10] also prove that this approach is theoretically complete as it can represent all material functions. However, in their work the modeling of the solid geometry and the material features by Euclidean distance fields is practically not considered.

### 2.3.2 Discussion

We propose to use the formulations for the set-theoretic operations and the SARDF framework introduced in Chapter 3, Section 3.1 to extend the hypervolume constructive framework discussed previously [69] for distance-based modeling. In the latter model, both the geometry of the solid, and the shape of the definition domains for the attributes can indeed be defined in constructive ways using SARDF primitives and operations.

The modified constructive hypervolume model is used to answer the question (section 5.2 of [10]) of the practical ways to compute the Euclidean distance field and then combined with the work of [69, 10] to model constructive heterogeneous objects using signed distance fields.

### 2.4 Automation of modeling and reverse engineering

Modeling requires lots of skills and can be a difficult and time-consuming task. We look at automating the modeling process. Practically automation is required when making models for existing objects digitized by a scanner device. This process, called
reverse engineering, consists usually in the following steps – not necessary in a linear order:

- data capture; the most famous method consists in using a laser scanner, but there are many other methods [107]
- preprocessing; like denoising, computing consistent and globally oriented normals, combining multiple views obtained from different data acquisitions
- segmentation and surface fitting; where data points are grouped into sets to which an appropriate surface is fitted
- geometric model creation

A detailed description of all these steps is outside the scope of this dissertation. Each of them contains several open problems.

Modeling automation is also required when converting data from a format to another, for example from BRep to CSG.

Generally, it is possible to distinguish between reverse engineering for computer graphics purposes and CAD purposes, because they have different goals, even if the global framework is the same. Reverse engineering for CAD purposes has stronger requirements for the generated CAD model. The following review of existing works follows this distinction.

2.4.1 Creation of triangle meshes and implicit surface fitting

Ohtake et al. [68] computes a mesh approximation of scattered point data by creating an adaptive sparse cover of the point-set, creating auxiliary points corresponding to the spheres, connecting these points, and finally filling the holes and cleaning the mesh.

**Implicit surface fitting:** Fitting implicit surfaces, or FRep, is a very popular method for generating surfaces from discrete point-sets. A mesh approximation of the surface can then be generated by using the marching cube [56], or other algorithms for polygonizing implicit surfaces, like [12, 71]. The reconstruction method proposed by Muraki [64] consists in fitting blobby models [11] to range data. Savchenko et al. [89] and later Turk et al. [105] proposed to fit a linear combination of radial basis functions to the point-set. Compactly supported radial basis functions were introduced by Morse et al. [63] to decrease the complexity in time and memory of the previous method. Partition of Unity was introduced by Ohtake et al. [66] as an elegant way to partition the point-set in order to decrease drastically the time and memory complexity.

These methods produce verbose models, even in the case of implicit surface fitting which may contain a lot of coefficients for the splines as well as the original point-set. They are also useless for inspection and reuse of the structure of the object in contrary to objects built using constructive geometry methods. Furthermore, besides the work of Amenta [2] they lack of theoretical guarantees: existence of cracks and holes, different topology than the original models.

These methods are however really good for producing nice looking triangle meshes to be used in various applications such as games, virtual museum and others. For complex freeform shapes with lots of small details, they appear also as the only practical methods.
2.4.2 Reverse engineering in CAD

Reverse engineering for CAD purposes has the same goal: to generate a CAD model for existing objects. It has however different criteria: the final created model should be a valid CAD model for solid modeling and ready to undergo further operations.

The main difference between the reverse engineering for CAD purposes and for computer graphics applications is that the creation of geometric models for CAD purposes concentrate on accurate and consistent models using standard surfaces as found in the common CAD/CAM systems [107]. The standard model representation is usually the boundary representation or BRep.

Parts of the problems are identifying sharp edges, treatment of blends, providing continuity and smoothness between the patches. Another important problem is the creation of geometric models respecting constraints such as for example: planes are parallel, spheres are concentric, and others [5].

The segmentation part, which is usually skipped in the reconstruction methods for computer graphics, is especially difficult. It consists in clustering the original point-set into subsets that correspond to some common primitives. It is a key step in identifying the logical structure of the final CAD model. The dissertation of Vanco [106] studies the problem of direct segmentation of point-set; direct segmentation means that no intermediate triangulation of the point-set is computed to do the segmentation. It relies on normal and curvature estimations, and segmentation by clustering the points using the normal and curvature information. Simple primitives are fitted using the information available from the normals and the curvature and help to produce a more robust segmentation.

The drawback of this approach is that a good computation of normals and curvature is difficult. Computing normals with a global consistent orientation is NP [43]. Hoppe [43] proposes a heuristic method to solve it. Computing numerically reliable curvature information is even more difficult [67, 110, 20].

Benko and Varady uses also a direct segmentation by computing a series of simple tests to split the original point-set [6]. Marshall et al. [57] try to fit common primitives
(sphere, torus, plane) by an approximation of the Euclidean distance.

After the segmentation of the point-set is segmented, primitives are associated and fitted to each subsets. Sometimes the fitting is part of the segmentation [57].

The final step is the creation of the CAD model. It consists in grouping the fitted primitives to make a valid BRep: most of the difficulties are from the constraints requirements of the final model, for example, planes are perpendicular, or parallel, smooth blend between parts, and others. Another problem is the creation of a valid BRep model, and we saw in Section 2.1 that it may be difficult to fulfill.

2.4.3 Boundary Representation to CSG conversion

An important problem for solid modeling, but generally true for many other branches of computer science and engineering, is the conversion of data from one representation to an other. The problem of BRep to CSG conversion is an important practical problem but unfortunately difficult to solve.

This problem is related with reverse engineering of solids. Suppose that you have been successful in generating a BRep model from a point-set, or even more simply that you have a segmentation of the point-set, with a primitive fitted to each of the subsets. Then you want to convert the BRep to a CSG representation or find a CSG representation that uses the fitted primitives.

The question of converting a BRep to a CSG representation has been firstly investigated by Rvachev et al. in the two-dimensional case [85], where an algorithm to convert a shape defined by a two-dimensional linear polygon to a CSG representation is described. In the English literature, a similar algorithm is attributed to Batchelor [4], other versions and improvements have been described in [112, 103, 23]. These algorithms are based on the concept of convex deficiency tree: any linear polygon can be represented by the difference between its convex hull and a finite number of concavities.

Shapiro extended the algorithm to handle curved polygons [93]. Similar algorithms were adapted to three-dimensional polyhedra [111], but unfortunately do not work
for any polyhedra.

In three-dimensional spaces, the problem has been solved for solids bounded by second degree surfaces [95, 17]. The algorithm may require some additional halfspaces not available from the face’s information, or the segmentation.

### 2.4.4 Discussion

Methods generating a triangle mesh approximation of the surface are unacceptable for most of the applications involving the reconstructed model. The generated triangle mesh provides no theoretical guarantee of accuracy or consistency. The same fact holds for methods based on fitting of implicit surfaces. In most of the cases, there is no proof of accuracy or consistency. For example: can we guarantee that there will be no extra isosurfaces, or extra isolated point-sets of lower dimension corresponding to the isovalue of interest?

Reconstructing objects defined by implicit surfaces seems preferable than reconstructing triangle meshes as it provides a function to define the object that can be used later as a valid FRep model. This function can be used for example to generate meshes adapted to the requirements of the finite element methods [49] or it can be used directly in mesh-free methods [34].

Reverse engineering in CAD has stronger requirements than reverse engineering with applications in computer graphics: it aims at providing valid, accurate and consistent models. However, the target model is a boundary representation model of the object, which is known to have various problems as seen in Section 2.1. For some applications, it is interesting to possess a constructive tree, because it can be used later to inspect or modify the structure of the recovered object.

The same requirement of getting a constructive tree can be made when fitting implicit surfaces. When recovering an FRep model from a given existing solid, a constructive object carries much more semantical information than a set of basis functions and their associated coefficients.
Getting a constructive tree gives the possibility to later treat and handle the object in the same way it could be done with any objects modeled by a user. Especially one can inspect the structure of the object, modify primitives, or operations, to create new models. The figure 2.6, left illustrates the importance of a constructive tree in the recovered model. Suppose we want to edit the model and create a new model with material added by blending as in Fig. 2.6, right; if the object is available as a constructive model, it is easy to simply replace the set-theoretic operation by its blending counterpart. It would be much more difficult to perform the same task with the boundary representation of the same object.

![Figure 2.6](image_url)

Figure 2.6: Left, A simple T like shape made by the union of two blocks. Right, the set-theoretic operations are replaced by their blending counterpart.

Being able to express the constructive tree as a real-valued function is a second important requirement. As highlighted before, a real-valued function can be used for further analytic treatment such as accurate meshing for finite element methods [49], or mesh-free methods [34].

Following these criteria, we consider the problem of reverse engineering objects by constructive FRep models. We introduce for that purpose the idea of parameterized template FRep. A parameterized FRep model, or template FRep model is a sketch made by a user, where the construction tree contains only specified operations and types of primitives, while the parameter values of operations and primitives are not
defined and recovered by fitting. These parameters have all a meaning and a semantic governing the shape of the solid. The modification of these parameters can result in various different shapes, which can be tuned to fit some modeling criteria.

Template models can exist in specialized libraries for each application domain: mechanical engineering, human prosthesis design, and others, or need to be created by a user. In a given domain of application, objects can have a similar shape that can be parameterized.

The creation of template models may still appear to be a difficult task. We look for the automation of this task by converting segmented point-sets to a constructive FRep. Parameters can be defined by the user or from statistical studies to generate a parameterized FRep from the later object.
Chapter 3

Theoretical approaches

In this chapter we present a theoretical approach to the open problems proposed in Introduction and in Section 2.

First, we propose a construction for new formulations of the set-theoretic operations intersection, union and difference [29, 31, 28] called Signed Approximate Real Distance Function or SARDF. These functions correspond to smooth min/max. We prove that the constructed functions are in $C^1$ over $\mathbb{R}^2 \setminus (0,0)$. The SARDF framework is then proposed and detailed: it consists in SARDF operations combined with rigid body motions and primitives defined by the distance function or an approximation. Algorithms for computing the distance to various primitives are given.

Modeling shapes in a constructive way can be a difficult and time consuming task, in order to automate this process, we introduce the concept of parameterized FRep model [26, 25]. We consider the meaning of a parametric family of functions where the parameters control the global shape of the final solid. The problem of tuning these parameters to fit a shape to a point cloud is considered. Fitting of parameterized models is used as a mean to automate modeling. We propose algorithms based on metaheuristic and direct methods to fit non-linear FRep models. These algorithms benefit from using the SARDF framework to define the parameterized FRep model.

In order to further automate the modeling process, we look at the problem of automatically generating a constructive FRep for solids defined by segmented point-sets. A genetic algorithm is proposed to solve this problem.
3.1 SARDF operations

R-functions have good properties of smoothness making them appreciated in shape and solid modeling. This smoothness is appreciated among others in material modeling, animation, or definition of blending. Unfortunately, the R-functions “destroy” quickly the distance properties of the argument functions. This effect was noticed by other researchers especially in the field of material modeling [97]. In contrary, min/max keeps a better approximation of the distance property for the constructive shape. However, it adds singularities to the constructed function in addition to the natural singularities of the true distance function.

The problem we address is to introduce new operations on functions corresponding to set-theoretic operations on solids such that these new operations have the following properties:

• they have better differential properties than min/max;

• they are a better approximation of the distance function than the R-functions and at least as good as min/max;

Some recent works proposed to modify the contour lines of the min and max functions in order to create some nice blending functions [44, 3]. The same techniques can be used to construct some smooth min/max: the sharp corners of the contour lines can be replaced by symmetric circular arcs, except the sharp corner passing through the origin, the radius of the circular arcs is either growing or bounded by two control lines, to better approximate the distance function. These smooth approximations are called SARDF operations, an abbreviation for Signed Approximate Real Distance Function.

We give the construction for different versions of the SARDF operations: at first a version with $C^1$ discontinuity on a circular arc is given; this formulation can be used in shape modeling, where the constraints on smoothness of the overall functions can be sometimes lower [29, 31]. When the smoothness is an important criterium, as
in some solid modeling operations, a different formulation is proposed [28]; its time complexity is however a bit higher.

We explain the construction of these functions in the following, together with their smoothness properties.

The constructions and properties of these functions are similar for the intersection and union as these operations present symmetries: \(\text{intersection}(x, y) = -\text{union}(-x, -y)\). We give constructions and properties only for the intersection. The difference is trivially obtained from the intersection by: \(S_1 \setminus S_2 = f_1 \wedge -f_2\).

### 3.1.1 A first approach: circular min approximation

In this section, we introduce a circular approximation of the min function for the set-theoretic operation intersection to approximate the signed real distance function. Any contour line of the min function has a sharp corner, corresponding to the union of two vertical and horizontal rays, see for instance Fig. 3.1 showing the sharp corners appearing when drawing different contour lines for the min function. This feature of the contour lines reflects the discontinuity of the partial derivatives of the min function that occurs at any point when two arguments are equal.

Following the general approach of [3, 44], we propose to replace the sharp corner in any contour line, except the contour line passing through the origin, with a circular arc. All operations are discussed for two halfspaces \(f_1 = x, f_1 \geq 0\) and \(f_2 = y, f_2 \geq 0\). We consider two straight lines, symmetric with respect to the line defined by \(y = x\) and with an angle \(\theta\) between these lines, which act as a frontier for the circular arcs. Applying these operations to arbitrary distance functions \(f_1\) and \(f_2\) consists in syntactically replacing \(x\) by \(f_1\) and \(y\) by \(f_2\).

The Euclidean plane is divided into four quadrants; the first quadrant corresponds to \(x > 0\) and \(y > 0\), the second quadrant to \(x < 0\) and \(y > 0\), the third quadrant to \(x < 0\) and \(y < 0\), and finally the fourth quadrant to \(x > 0\) and \(y < 0\). In the second and fourth quadrants, the approximate function for min is equal exactly to min; thus we restrict the discussion to the first and third quadrants, where the sharp corners
need to be smoothed.

**Circular min approximation: quadrant I**

We discuss here the circular approximation of the function $F(x, y) = \min(x, y)$ in the first quadrant, where $x > 0$ and $y > 0$. We want to replace any contour lines $F = d$ with a circular arc and two rays tangentially attached to it as shown in Fig. 3.2. The angle $\theta$ made by two straight lines $L_1$ and $L_2$ is introduced as in Fig. 3.3.

These two straight lines $L_1$ and $L_2$ break this first quadrant into three zones: A (below $L_1$), B (between $L_1$ and $L_2$) and C (above $L_2$), as shown in Fig. 3.3.

The attachment points $P_1$ and $P_2$ of the arc and the rays are placed on the lines $L_1$ and $L_2$ correspondingly. Figure 3.4 shows such a contour line configuration.

We are interested in the contour lines $\tilde{F} = d$ of the smooth approximation $\tilde{F}$ of the min function. Given an arbitrary point $P = (x, y)$, we need to calculate a function value $d$ for it.
Figure 3.2: Contour line of the function min has a sharp corner (left) to be replaced by a circular arc (right).

Figure 3.3: Two straight lines with the angle $\theta$ between them break the first quadrant into three zones.
Figure 3.4: New contour line configuration: the two rays are attached to the circular arc at the junction points $P_1$ and $P_2$. 
In zone A, $\bar{F}$ is equal to $\min(x, y)$, therefore the contour is a horizontal line going through the point $P$ and defined as $\bar{F} = y$. In zone C, $\bar{F}$ is also equal to $\min(x, y)$, so the contour is a vertical line going through the point $P$ and defined as $\bar{F} = x$.

Finally, in zone B, we want to have a circular arc passing through the point $P = (x, y)$. This arc should go through the point $P$ and change into the horizontal ray in zone A and into the vertical ray in zone C. Both of these rays are at the distance $d$ from the corresponding $x$ and $y$ axes. Such a distance is used for the definition of the value of the function. In order to calculate this distance $d$, we start from the equation of the circle passing through $P$:

$$(x - x_0)^2 + (y - y_0)^2 = R^2 \quad (3.1.1)$$

In this equation, $x_0$, $y_0$ and $R$ are unknown but can be expressed in terms of the value $d$ being searched, and $\alpha$, the angle between the straight lines and the axes. Figure 3.5 shows the unknowns and their geometric relations.

Figure 3.5: Unknowns of the Eq. 3.1.1 and their geometric relations.

First, $\alpha$ is expressed using $\theta$ (a parameter left to the user, expressing the angle between $L_1$ and $L_2$): $\alpha = \frac{(\frac{\pi}{2} - \theta)}{2}$. Then, from the lower triangle in zone A (Fig.
3.5), \( x_0 = d \tan(\alpha) \). By analogy, from the upper triangle in zone C (Fig. 3.5), \( y_0 = d \tan(\alpha) \), and \( R = x_0 - d \). By replacing these variables in Eq. 3.1.1, we obtain the following quadratic equation for the variable \( d \):

\[
d^2 \left[ \cotan^2(\alpha) + 2 \cotan(\alpha) - 1 \right] - 2 \left( x + y \right) \cotan(\alpha) + x^2 + y^2 = 0 \quad (3.1.2)
\]

The solution of Eq. 3.1.2 for the unknown \( d \) is:

\[
d = \begin{cases} 
-\frac{b \pm (b^2 - 4ac)^{0.5}}{2a} & \text{if } a \neq 0 \text{ and in zone B} \\
-\frac{c}{b} & \text{if } a = 0 \text{ and in zone B}
\end{cases}
\]

where \( a = \cotan^2(\alpha) + 2 \cotan(\alpha) - 1 \), \( b = -2 \left( x + y \right) \cotan(\alpha) \) and \( c = x^2 + y^2 \) are the coefficients of Eq. 3.1.2.

The final expression for the value of \( \tilde{F} \), at \( P \) in the quadrant I, is summarized below:

\[
\tilde{F}(P) = d = \begin{cases} 
-\frac{b \pm (b^2 - 4ac)^{0.5}}{2a} & \text{if } a \neq 0 \text{ and } P \text{ in zone B} \\
-\frac{c}{b} & \text{if } a = 0 \text{ and } P \text{ in zone B} \\
y & \text{if } P \text{ in zone A} \\
x & \text{if } P \text{ in zone C}
\end{cases}
\]

where \( a = \cotan^2(\alpha) + 2 \cotan(\alpha) - 1 \), \( b = -2 \left( x + y \right) \cotan(\alpha) \) and \( c = x^2 + y^2 \), and \( \alpha \) is an angle between \( L_1 \) and \( x \)-axis, and between \( L_2 \) and \( y \)-axis.

**Circular min approximation: quadrant III**

We consider now the approximation \( \tilde{F} \) of the min function in the third quadrant, where \( x < 0 \) and \( y < 0 \). The method is the same as for the first quadrant. The continuation of the two straight lines \( L_1 \) and \( L_2 \) into the third quadrant, breaks it into three zones: \( D \) above \( L_1 \), \( E \) between \( L_1 \) and \( L_2 \), and \( F \) below \( L_2 \) as indicated in Fig. 3.6.

Given an arbitrary point \( P = (x, y) \) in the third quadrant, we want to evaluate \( \tilde{F} \), the smooth approximation of the min function at this point.

In zone D, \( \tilde{F} \) is equal to \( \min(x, y) \), therefore the contour is a vertical line going through the point \( P \) and defined as \( \tilde{F} = x \). In zone F, \( \tilde{F} \) is also equal to \( \min(x, y) \), so the contour is a horizontal line going through the point \( P \) and defined as \( \tilde{F} = y \).
Figure 3.6: The continuation of $L_1$ and $L_2$ from the quadrant 1 ($x > 0$ and $y > 0$) breaks the quadrant 3 ($x < 0$ and $y < 0$ into three new zones $D$, $E$, and $F$.

In zone E, we want to have a circular arc passing through the point $P(x, y)$. This arc should go through the point $P$ and change into the horizontal ray in zone $F$ and into the vertical ray in zone $D$. Both of these rays are at the distance $d$ from the corresponding $x$ and $y$ axes. Such a distance is used for the definition of the value of the function.

Again, in order to calculate this distance $d$, we start from the equation of the circle passing through $P$:

$$\left(x - x_0\right)^2 + \left(y - y_0\right)^2 = R^2 \tag{3.1.3}$$

In this equation, $x_0$, $y_0$ and $R$ are unknown but can be expressed in terms of $d$ and angle $\alpha$. Fig. 3.7 shows the unknowns and their geometric relations.

First, $\alpha$ is expressed using $\theta$ (the angle between $L_1$ and $L_2$): $\alpha = \frac{(\frac{\pi}{2} - \theta)}{2}$. Then, from the triangle in zone $D$, $y_0 = d \tan(\alpha)$. By analogy, from the triangle in zone $F$, $x_0 = d \tan(\alpha)$, and $d = -(R + |x_0|)$. By replacing these variables in Eq. 3.1.3, we
obtain the quadratic equation for the variable $d$:

$$d^2 \left[ \tan^2(\alpha) + 2 \tan(\alpha) - 1 \right] - 2d \left( x + y \right) \tan(\alpha) + x^2 + y^2 = 0 \quad (3.1.4)$$

The solution of Eq. 3.1.4 for the unknown $d$ is:

$$d = \begin{cases} 
\frac{-b \pm (b^2 - 4ac)^{0.5}}{2a} & \text{if } a \neq 0 \text{ and } P \text{ in zone E} \\
-\frac{c}{b} & \text{if } a = 0 \text{ and } P \text{ in zone E}
\end{cases}$$

where $a = \tan^2(\alpha) + 2 \tan(\alpha) - 1$, $b = -2 \left( x + y \right) \tan(\alpha)$ and $c = x^2 + y^2$ are the coefficients of Eq. 3.1.4.

The final expression for the value of $\tilde{F}$, at $P$ in the quadrant III, is summarized below:
where $a = \tan^2(\alpha) + 2 \tan(\alpha) - 1$, $b = -2 (x + y) \tan(\alpha)$ and $c = x^2 + y^2$.

**Intersection for two given shapes**

The intersection for two given shapes, defined by the signed distance functions $f_1$ and $f_2$ is obtained by replacing $x$ and $y$ in the above equations by $f_1$ and $f_2$.

**Problem of the circular approximation**

The use of the described above circular approximations for the min and max functions can provide the $C^1$ approximation of the resulting distance function for constructive shapes built using normalized primitives (defined by distance functions). Unfortunately, this approach has the following problem: the radius of the circular arc used to replace the sharp corners in the contour lines keeps growing with the distance from the initial surfaces. Figure 3.8 illustrates this problem for the case of the min function approximation. Because of this behavior of the arc radius, the error of the distance function approximation grows infinitely with the distance.

We propose to prevent the radius from growing infinitely by introducing a fixed radius circular arc, and by switching to it, when some threshold for the radius is reached.

**3.1.2 Circular min approximation with bounded radius**

In the previous approach, the smoothing area grows infinitely in size inside the given angle. It is better to introduce a fixed threshold for the radius $R$ of the circular arc. Consider for example the distance to a L shape: in the quadrant I, an unbounded circular approach gives an unbounded grow of the error between the distance and the constructed function. Note that in the quadrant III, for that case, an unbounded
Figure 3.8: The radius of the circular arc is growing with the distance from the origin, thus increasing the error in the distance function made by the approximation.

The approach corresponds to the best solution. A new bounding band can be introduced by two parallel straight lines that enclose the arcs with the fixed radius. These band lines are defined by a shift of the line $y = x$ at $R$ distance in positive and negative $x$ directions: $y = x - R$ and $y = x + R$.

The principle of the circular approximation is illustrated in Fig. 3.9 by the contour lines of the circular max approximation (right) and the circular min approximation (left).

Figure 3.9: Contour lines of the circular min (left) and max (right) approximations; note the use of different threshold radius.
Approximate min: quadrant I

The intersections of the two parallel bounding band lines, \( y = x + R \) and \( y = x - R \), with the lines parallel to the axes: \( x = R \) and \( y = R \), result in two points: \( \mathbf{A}_1 = (2R, R) \) and \( \mathbf{A}_2 = (R, 2R) \), as shown in Fig. 3.10. These points are connected by the circular arc: \((x - 2R)^2 + (y - 2R)^2 = R^2\), with \( x < 2R \) and \( y < 2R \). This makes a natural boundary, that splits the first quadrant into two zones I and II, for applying two approaches for the approximation (see Fig. 3.10).

Figure 3.10: The first quadrant is divided into two zones. The circular approximation is applied in zone I, whereas we introduce a fixed radius approximation with the bounding band in zone II.
Zone I corresponds to the set of points \( P = (x, y) \) for which \( x < R \) or \( y < R \) or \( (x < 2R \text{ and } y < 2R \text{ and } (x-2R)^2 + (y-2R)^2 > R^2) \), see Fig. 3.10. In this zone, the circular approximation described in previous subsection is used. In this case, the bounding lines for the angle \( \theta \) are:

- \( L_1: O - A_1 \), where \( O \) is the origin (0,0),
- \( L_2: O - A_2 \).

and the angle \( \alpha \) is defined by \( \cotan(\alpha) = 2 \). We remind the expression of the circular approximation \( \tilde{F} = d \) for the min function, in the first quadrant:

\[
\tilde{F}(P) = d = \begin{cases} 
\frac{-b \pm (b^2 - 4ac)^{0.5}}{2a} & \text{if } P \text{ in zone B} \\
y & \text{if } P \text{ in zone A} \\
x & \text{if } P \text{ in zone C}
\end{cases}
\]

where \( a = \cotan^2(\alpha) + 2 \cotan(\alpha) - 1 = 7, \ b = -2 \ (x + y) \cotan(\alpha) = -4 \ (x + y) \) and \( c = x^2 + y^2 \).

Zone II In the zone II, the fixed radius with the bounding band is applied to get a smooth approximation \( \tilde{F} \) of the min function. Outside the bounding band lines \( y = x + R \) and \( y = x - R \), the approximation is min itself, therefore we take \( \tilde{F} = \text{min}(x, y) \). Between the lines, we start from the equation of the circle: \((x - x_0)^2 + (y - y_0)^2 = R^2\), where \( x_0, y_0 \) are parameters that can be expressed using the radius \( R \) and the distance \( d \), which is the searched value of \( \tilde{F} \).

Figure 3.11 displays these different parameters and their geometric relations. It is obvious that \( x_0 = y_0 = d + R \). Replacing \( x_0 \) and \( y_0 \) in the equation of the circle, the following quadratic expression of the unknown \( d \) is obtained: \( 2d^2 + d (-2x - 2y + 4R) + (x^2 + y^2 - 2xR - 2yR + R^2) = 0 \). The solution for the unknown \( d \), gives the searched value for the smooth approximation of the min function: \( d = \frac{-b \pm (b^2 - 4ac)^{0.5}}{2a} \), where \( a = 2, \ b = -2x - 2y + 4R \) and \( c = x^2 + y^2 - 2xR - 2yR + R^2 \) are the coefficients of the quadratic equation.
Figure 3.11: Geometric relations between the parameters of the circular approximation with the bounding band in zone II.
The expression for $\tilde{F}$ in the zone II becomes:

$$\tilde{F}(P) = d = \begin{cases} 
\frac{-b\pm(b^2 - 4ac)^{0.5}}{2a} & \text{inside the bounding band} \\
y & \text{below } y = x - R \\
x & \text{above } y = x + R 
\end{cases}$$

where $a = 2$, $b = -2x - 2y + 4R$ and $c = x^2 + y^2 - 2xR - 2yR + R^2$.

**Final expression for the intersection in the quadrant I**

We give the final expression for the intersection $\tilde{F}$ approximating the min function in the first quadrant. Given a point $P = (x, y)$ in the first quadrant, the value $d$ of $\tilde{F}$ at $P$ is given by:

$$\tilde{F}(P) = d = \begin{cases} 
\frac{-b_1\pm(b_1^2 - 4a_1c_1)^{0.5}}{2a_1} & \text{if } P \text{ in zone I, B} \\
y & \text{if } P \text{ in zone I, A} \\
x & \text{if } P \text{ in zone I, C} \\
\frac{-b_2\pm(b_2^2 - 4a_2c_2)^{0.5}}{2a_2} & \text{if } P \text{ in zone II, and inside the bounding band} \\
y & \text{if } P \text{ in zone II and below } y = x - R \\
x & \text{if } P \text{ in zone II and above } y = x + R 
\end{cases}$$

where $a_1 = \cotan^2(\alpha) + 2 \cotan(\alpha) - 1 = 7$, $b_1 = -2(x + y) \cotan(\alpha) = -4(x + y)$, $c_1 = x^2 + y^2$, $a_2 = 2$, $b_2 = -2x - 2y + 4R$ and $c_2 = x^2 + y^2 - 2xR - 2yR + R^2$.

**Approximate min: quadrant III**

We consider now the same approach to approximate the min function with the bounding band in the third quadrant. The intersections of the two parallel bounding band lines: $y = x + R$ and $y = x - R$, with the lines parallel to the axes $x = -R$ and $y = -R$, result in two points: $A_3 = (-2R, -R)$ and $A_4 = (-R, -2R)$, as shown in Fig. 3.12. These points are connected by the circular arc: $(x + R)^2 + (y + R)^2 = R^2$, with $x < -R$ and $y < -R$. This makes a natural boundary, that splits the third quadrant into two zones III and IV, for applying the two different approaches for the approximation (see Fig. 3.12).
Figure 3.12: The third quadrant is divided into two zones. The circular approximation is applied in zone III, whereas we introduce a fixed radius approximation with the bounding band in zone IV.
Zone III  Zone III corresponds to the set of points \( P = (x, y) \) for which \( x > -R \) or \( y > -R \) or \( (x > -2R \) and \( y > -2R \) and \( (x + R)^2 + (y + R)^2 < R^2 \)), see Fig. 3.12. In this zone, the circular approximation described above is applied. In this case, the bounding lines for the angle \( \theta \) are:

- \( L_1: O - A_4 \), where \( O \) is the origin \((0,0)\),
- \( L_2: O - A_3 \).

and the angle \( \alpha \) is defined by \( \cotan(\alpha) = 2 \). We remind the expression of the smooth approximation \( \tilde{F} = d \) for the min function, in the third quadrant:

\[
\tilde{F}(P) = d = \begin{cases} 
-\frac{-b \pm (b^2 - 4ac)^{0.5}}{2a} & \text{if } P \text{ in zone E} \\
y & \text{if } P \text{ in zone F} \\
x & \text{if } P \text{ in zone D}
\end{cases}
\]

where \( a = \tan^2(\alpha) + 2 \tan(\alpha) - 1 = \frac{1}{4} \), \( b = -2 (x + y) \tan(\alpha) = -(x + y) \) and \( c = x^2 + y^2 \).

Zone IV  In zone IV, we switch to a fixed radius approximation with the bounding band. Outside the bounding band, we take \( \tilde{F} = \min(x, y) \). Inside the bounding band, we start from the equation of the circle: \( (x - x_0)^2 + (y - y_0)^2 = R^2 \), where \( x_0 \), \( y_0 \) are parameters that can be expressed using the radius \( R \) and the distance \( d \), which is the searched value of \( \tilde{F} \).

Figure 3.13 shows these different parameters and their geometric relations. It is obvious that \( |x_0| = |y_0| = |d| - R \). Replacing \( x_0 \) and \( y_0 \) in the equation of the circle, the following quadratic expression of the unknown \( d \) is obtained: \( 2d^2 + d (-2x - 2y + 4R) + (x^2 + y^2 - 2xR - 2yR + R^2) = 0 \). The solution for the unknown \( d \) gives the value \( d = -\frac{-b \pm (b^2 - 4ac)^{0.5}}{2a} \), where \( a = 2 \), \( b = -2x - 2y + 4R \) and \( c = x^2 + y^2 - 2xR - 2yR + R^2 \) are the coefficients of the quadratic equation.
The expression for $\tilde{F}$ in zone IV becomes:

$$\tilde{F}(P) = d = \begin{cases} 
-b \pm (b^2 - 4ac)^{0.5} 
& \text{inside the bounding band} \\
2a 
& \text{below } y = x - R \\
x 
& \text{above } y = x + R 
\end{cases}$$

where $a = 2$, $b = -2x - 2y + 4R$ and $c = x^2 + y^2 - 2xR - 2yR + R^2$.

**Final expression for the intersection in the quadrant III** We give the final expression for the intersection $\tilde{F}$ approximating the min function in the third quadrant. Given a point $P = (x, y)$ in the third quadrant, the value $d$ of $\tilde{F}$ at $P$ is defined
as:

$$\tilde{F}(P) = d = \begin{cases} 
\frac{-b_1 \pm (b_1^2 - 4a_1c_1)^{0.5}}{2a_1} & \text{if } P \text{ in zone III, } E \\
y & \text{if } P \text{ in zone III, } F \\
x & \text{if } P \text{ in zone III, } D \\
\frac{-b_2 \pm (b_2^2 - 4a_2c_2)^{0.5}}{2a_2} & \text{if } P \text{ in zone IV, and inside the bounding band} \\
y & \text{if } P \text{ in zone IV and below } y = x - R \\
x & \text{if } P \text{ in zone IV and above } y = x + R 
\end{cases}$$

where \( a_1 = \tan^2(\alpha) + 2 \tan(\alpha) - 1 = \frac{1}{4} \), \( b_1 = -2(x + y)\tan(\alpha) = -(x + y) \), \( c_1 = x^2 + y^2 \), \( a_2 = 2 \), \( b_2 = -2x - 2y + 4R \) and \( c_2 = x^2 + y^2 - 2xR - 2yR + R^2 \).

**Problems with the bounded circular approach**

The main problem with this approach is that the function has \( C^1 \) discontinuities in the point \((0,0)\) and also on the curves defined by the circular arc boundaries \( A_1 - A_2 \) and \( A_3 - A_4 \) (see Fig. 3.10, 3.12). The latter discontinuities on the arcs come from the fact that the radius is first growing linearly and then remains constant. The \( C^1 \) discontinuity in \((0,0)\) is used to model the sharp corners and edges in the geometric modeling space and is not really problematic but rather a modeling feature; the other curves of \( C^1 \) discontinuity may pose problems in modeling and applications. In some cases, like for blending or shape operations, it may practically not be a problem [31] because the problematic points are well located and can be avoided during the modeling of the shape, but in engineering applications, such as for example material modeling, smoothness of the overall function is often required [10, 28].

\( C^1 \) continuity can be obtained by replacing the lines delimiting the linear growing of the radius by arc of parabolas.

**3.1.3 SARDF intersection construction**

The sharp corners, in quadrant 1 and 3, are replaced as above by circular arcs with growing radius. We propose two approaches to control the growth of the radius
in quadrant 3: in the first one, the radius is bounded by two straight lines after a threshold radius is reached, whereas the radius is allowed to grow infinitely in the other. In quadrant 1, the growth of the radius is always bounded after some threshold. The idea behind these strategies is to mimic at best the behaviour of the distance function. The general idea is illustrated in Fig. 3.14 for the two different approaches.

Figure 3.14: Illustration of the general idea for the construction of SARDF intersection. The two possible approaches in the quadrant 3 are given left and right. Left, the growth of the radius is bounded by two straight lines; right, the growth of the radius is unbounded.

**Quadrant 1**

In quadrant 1, the sharp corners of the intersection operation found in every contour lines are replaced by a circular arc with growing radius. This approach is illustrated Fig. 3.15. Two parabola symmetric by $y = x$ are used to delimit the circular arc approximation (Zone I,B). The growth of the circular arc is bounded by introducing a threshold radius $R$ (Zone II). A bounding band is introduced by two parallel straight lines that enclose the circular arc with fixed radius. These band lines are defined by a shift of the line $y = x$ at $R$ distance in positive and negative $x$ directions. The two branches of the parabolas are defined to be tangent to the two parallel lines $y = x - R$ and $y = x + R$ at the connecting points $(R, 2R)$ and $(2R, R)$ and pass through the
origin \((0,0)\), it gives the expressions for these two parabolas: \(y = \frac{x^2}{4R}\) and \(x = \frac{y^2}{4R}\). Note that the use of parabolas to restrict the circular approximation ensures that the constructed function is \(C^1\) on the arc of circle \(A_1A_2\).

![Figure 3.15: The different zones of the SARDF intersection in quadrant 1](image)

**Zone I,B**  Given a point \((x, y)\) in the first quadrant, Zone I, B, we want to calculate the iso-level value \(d\) for the SARDF intersection at this point. It belongs to a circular arc that is tangentially connected to two horizontal and vertical rays when reaching the parabola (see Fig. 3.15). The equation of this arc is \((x - x_0)^2 + (y - y_0)^2 = r^2\), where \(x_0, y_0\) and \(r\) need to be expressed as functions of the searched value \(d\). The point at the intersection of the parabola and the iso-level \(d\) of the searched function, is at a distance \(d\) from the axis \(y = 0\). Because this point belongs also to the parabola, it satisfies \(d = \frac{x_0^2}{4R}\). By symmetry it comes that: \(d = \frac{y_0^2}{4R}\).
The coordinates of the center of the circular arc \((x_0, y_0)\) satisfy the following equality: \(x_0 = y_0 = d + r\), it follows that \(r = y_0 - d = 2\sqrt{Rd} - d\). By plugging everything in the equation of the circle, using the substitution of variables \(\sqrt{d} = z\) and expanding, we obtain the following algebraic equation of degree four in \(z\):

\[
z^4 - 4\sqrt{R}z^3 - 4Rz^2 + 4\sqrt{R}(x + y)z - (x^2 + y^2) = 0 \quad (3.1.5)
\]

Thus in the first quadrant, in the zone I, B, the expression of the intersection is the square of one of the four roots of the algebraic equation 3.1.5. Roots of algebraic equation of degree four are known algebraically. The root of interest is found by using one of the limit conditions, for example \(d(2R, R) = R\).

**Zone II, inside the bounding band** Given a point \((x, y)\) in zone II, within the bounding band (see Fig. 3.15), we want to compute the iso-level value \(d\) of the SARDF intersection at that point. This point belongs to a circular arc that is tangentially connected to two horizontal and vertical rays when reaching the two lines of the bounding band. The equation of this circular arc is: \((x - x_0)^2 + (y - y_0)^2 = R^2\). This time \(R\) is constant, thus only \(x_0\) and \(y_0\) need to be expressed as functions of \(d\).

The coordinates \((x_0, y_0)\) of the circular arc satisfy: \(x_0 = y_0 = d + R\). After substitution into the equation of the circular arc and expanding this equation, \(d\) is one of the two solutions of the following second degree algebraic equation:

\[
2d^2 + d(4R - 2x - 2y) + (x^2 + y^2 - 2R(x + y) + R^2) = 0 \quad (3.1.6)
\]

The root of interest is obtained by using the limit condition: \(d(2R, R) = R\).

**Zone I, A and C, and II outside the bounding band** The function behaves exactly like min.

**Quadrant 3**

In quadrant 3, we consider two possible approaches: one is similar to the approach detailed above, the second uses two lines symmetric by \(y = x\) and opened by an angle
$\theta$. The construction in the first approach is similar to the above detailed construction. We give some details for the second approach, illustrated Fig. 3.16.

Figure 3.16: The different zones in the second possible approach in the quadrant 3

Considering a point $(x, y)$ in the quadrant, if it is outside the angle sector made by the two lines (Fig. 3.16), then the value of the function is exactly given by min. If it is inside, then the value of the function is computed as follows. The point belongs to the circular arc given by: $(x - x_0)^2 + (y - y_0)^2 = r^2$. $x_0, y_0$ and $r$ are expressed as functions of $d$, the searched function value, and $\alpha$ the line-axis angle (see Fig. 3.16). It comes that $d$ is the solution of:

$$d^2(tan^2(\alpha) + 2tan(\alpha) - 1) - 2d(x + y)tan(\alpha) + x^2 + y^2 = 0 \quad (3.1.7)$$

verifying $d(x, \frac{x}{tana}) = \frac{x}{tana}$ (for example).

Quadrants 2 and 4

In quadrants 2 and 4, the smooth min behaves exactly like min.
3.1.4 Smoothness of the SARDF intersection

The SARDF operations are functions in $C^1(\mathbb{R}^* \times \mathbb{R}^*)$; consequently the singularities in the resulting function are due only to: the primitives or the sharp features in the resulting solid. The discontinuity of the gradient in the origin is intentional to allow creation of sharp features.

The main steps of the proof are given for the SARDF intersection only. In quadrants 2 and 4 the function is trivially $C^1$. In quadrants 1 and 3 the function is symmetric with respect to the line $y = x$. Only the subset below this line needs to be studied. The function is also trivially $C^1$ piecewise, the discontinuity (of the function values or the derivatives values) can appear only at the boundaries between the different expressions: on the branches of the parabola, the straight lines ($y = x - R$ and $y = x + R$) or the arc boundary $A_1A_2$ between the growing radius and the fixed radius (quadrants 1 or 3), or on the straight lines (quadrant 3).

$C^1$ continuity in quadrant 1

The SARDF intersection is piecewise continuous. The continuity on the parabola branches, and on the lines is obvious by construction. We need to check it only on the circular arc boundary $A_1A_2$.

Continuity at the circular arc boundary $A_1A_2$ We study the continuity of the expression of the SARDF intersection at the circular arc boundary between the zones I,B, and II. Let $P = (x, y) = (2R + R\cos(u), 2R + R\sin(u))$, with $u \in \left[\frac{3\pi}{4}, \frac{5\pi}{2}\right]$, be a point on that circular arc. The value of the function at $P$ is given by its value at the point $A_1$ and is $y = R$. We check that this value matches the value of the expressions in the zones I,B and II at $P$, by checking that the algebraic equations 3.1.5 and 3.1.6 hold. After applying some calculus, we confirm that the relations given by Eq. 3.1.5 and 3.1.6 hold and we can conclude with the continuity at that boundary.
We give implicit definitions for the partial derivatives of the smooth function \(d\) in the quadrant 1, and then prove it is \(C^1\) on that domain, with exception of the origin, by verifying that the partial derivatives match on the boundaries points.

**Expression for the partial derivatives in zone I,B:** In zone I,B, the square root of the function \(\sqrt{d}\) satisfies the algebraic equation 3.1.5. Taking the partial derivative by \(x\) of 3.1.5 gives: \(4z_xz^3 - 12\sqrt{R}z_xz^2 - 8Rz_xz + 4\sqrt{R}(x + y)z_x + 4\sqrt{R}z - 2x = 0\). It follows that: 
\[
z_x = \frac{2x - 4\sqrt{R}z}{4z^3 - 12\sqrt{R}z^2 - 8Rz + 4\sqrt{R}(x + y)}.
\]
Since \(z = \sqrt{d}\), it comes that \(z_x = \frac{1}{2}d_x \frac{1}{\sqrt{d}}\). Combining it with the previous expression, we get a relation for the partial derivative of the function in zone I,B
\[
d_x = 2\sqrt{d} \frac{2x - 4\sqrt{R}z}{4z^3 - 12\sqrt{R}z^2 - 8Rz + 4\sqrt{R}(x + y)}
\]
(3.1.8)
with \(z(x, y) = \sqrt{d(x, y)}\).

By the same procedure, we obtain an expression for the partial derivative by \(y\) in zone I,B:
\[
d_y = 2\sqrt{d} \frac{2y - 4\sqrt{R}z}{4z^3 - 12\sqrt{R}z^2 - 8Rz + 4\sqrt{R}(x + y)}
\]
(3.1.9)
with \(z(x, y) = \sqrt{d(x, y)}\).

**Expression for the partial derivatives in zone II, within the bounding band:**

In zone II, within the bounding band, the function satisfies Eq. 3.1.6. With the same method as above, we take the partial derivative by \(x\), it gives: \(4d_xd + d_x(4R - 2x - 2y) - 2d + (2x - 2R) = 0\). It follows that:
\[
d_x = \frac{2d - 2(x - R)}{4d + (4R - 2x - 2y)}
\]
(3.1.10)

Similarly an expression for the partial derivative by \(y\) can be obtained:
\[
d_y = \frac{2d - 2(y - R)}{4d + (4R - 2x - 2y)}
\]
(3.1.11)
Expression for the partial derivatives in zone I, A and zone II, below $y = x - R$. The function is exactly min in these two areas, and so the partial derivatives in the $x$ direction is 0 and 1 in the $y$ direction.

We verify the continuity of the partial derivatives at the different boundary curves.

**Continuity of the partial derivatives at the parabolic arc** Let $\mathbf{P} = (x, y) = (u, \frac{u^2}{4R})$, with $u \in [0, 2R]$, be a point on the arc, at $\mathbf{P}$, $z = \sqrt{d} = \frac{u}{2\sqrt{R}}$; it is easy to verify that at $\mathbf{P}$ equations 3.1.8 and 3.1.9 give: $d_x = 0$ and $d_x = 1$.

**Continuity of the partial derivatives at the line $y = x - R$** Let $\mathbf{P} = (x, y) = (u, u - R)$, with $u \in [2R, \infty]$ be a point on the line, at $\mathbf{P}$, $z = \sqrt{d} = u - R$; it is easy to verify that at $\mathbf{P}$ equations 3.1.10 and 3.1.11 give: $d_x = 0$ and $d_x = 1$.

**Continuity of the partial derivatives at the circular arc boundary** Let $\mathbf{P} = (x, y) = (2R + R\cos(u), 2R + R\sin(u)), u \in \left[\frac{5\pi}{4}, \frac{3\pi}{2}\right]$ be a point on the circle boundary between the growing radius zone and the constant radius zone. At $\mathbf{P}$, the value of the function is $R$. Using these informations in equation 3.1.8, we obtain after straightforward calculus $d_x = \frac{\cos(u)}{\cos(u) + \sin(u)}$. Similarly, with equation 3.1.10: $d_x = \frac{\cos(u)}{\cos(u) + \sin(u)}$.

Similarly for the partial derivative $d_y$, equation 3.1.9 gives: $d_y = \frac{\sin(u)}{\cos(u) + \sin(u)}$. And equation 3.1.11 gives: $d_y = \frac{\sin(u)}{\cos(u) + \sin(u)}$.

With the equality of the partial derivatives we can conclude to the $C^1$ continuity of the SARDF intersection on the boundary circle. The $C^1$ continuity still holds at $A_1$, with $u = \frac{3\pi}{2}$.

**$C^1$ continuity in quadrant 3**

There are two possible approaches in the quadrant 3 to constructing the SARDF intersection. One is similar to the construction in quadrant 1, with a growing radius smoothing bounded by a threshold. The proof for the smoothness of the function is
similar as in quadrant 1. We give a sketch of the proof for the second approach. The function is continuous by construction and piecewise $C^1$, except at the origin. We check only the continuity of the different partial derivatives at the boundary line.

We use equation 3.1.7 to have expressions of the partial derivatives inside the two boundary lines.

$$d_x(x, y) = \frac{2d(x, y)\tan(\alpha) - 2x}{2d(x, y)(\tan^2(\alpha) + 2\tan(\alpha) - 1) - 2(x + y)\tan(\alpha)}$$

(3.1.12)

$$d_y(x, y) = \frac{2d(x, y)\tan(\alpha) - 2y}{2d(x, y)(\tan^2(\alpha) + 2\tan(\alpha) - 1) - 2(x + y)\tan(\alpha)}$$

(3.1.13)

It is easy to check that the partial derivatives are continuous at any point on the boundary line, by evaluating $d_x$ and $d_y$ at a point of the line and comparing with the values of the partial derivatives from the adjacent zone (where the function behaves like min, thus has partial derivatives with values 0 along $x$ and 1 along $y$). Given $P : (u, \frac{u}{\tan(\alpha)}), u \in \mathbb{R}^-$ a point on the lower boundary line, we have $d_x(P) = 0$ and $d_y(P) = 1$.

3.1.5 SARDF framework

The SARDF framework is a restriction of the FRep system where the SARDF implementations of the set-theoretic operations are used instead of the R-functions or min/max and primitives are limited to distance functions or approximations. Of course, primitives, which are not defined by distance functions or an approximation, can be used, as well as operations, which do not conserve the distance property or a reasonable approximation, however it will result in a global defining function for the object, which do not keep the Euclidean distance property.

SARDF operations

SARDF operations: intersection, difference, and union of two geometric objects defined by distance functions $f_1$ and $f_2$ are trivially obtained by replacing syntactically $x$ and $y$ in the previous SARDF functions by $f_1$ and $f_2$. Note that $x$ and $y$ can be seen
as two real functions \((x, y, z) \rightarrow x\) and \((x, y, z) \rightarrow y\), which correspond geometrically to two orthogonal halfspaces.

The smoothness of the resulting function defining the final object depends of the smoothness of the SARDF functions, which was studied above, and of the primitives. The gradient of the function resulting from the application of a SARDF operation \(F\) to two primitives \(f_1\) and \(f_2\) is given by:

\[
\nabla F = \left( \frac{\partial F}{\partial f_1} \frac{\partial f_1}{\partial x} + \frac{\partial F}{\partial f_2} \frac{\partial f_2}{\partial x} \right)
\]

\[
\frac{\partial F}{\partial f_1} \frac{\partial f_1}{\partial y} + \frac{\partial F}{\partial f_2} \frac{\partial f_2}{\partial y}
\]

\[
\frac{\partial F}{\partial f_1} \frac{\partial f_1}{\partial z} + \frac{\partial F}{\partial f_2} \frac{\partial f_2}{\partial z}
\]

(3.1.14)

\(C^1\) discontinuity of the arguments of the SARDF operations will result in \(C^1\) discontinuity of the resulting composed functions.

**SARDF primitives**

As mentioned earlier, we are interested only in primitives for which an expression (closed-form), an evaluation (numerical) procedure or an approximation for the Euclidean distance from the current point to the surface is available.

**Quadrics:** It is possible to have analytical expressions for all the natural quadrics. It should be noticed that natural quadrics are the main primitives of most of the constructive solid geometry (CSG) solid modeling kernels, and are enough to model most of the objects at least in mechanical engineering.

Quadric primitives can be classified into two groups:

- primitives with analytical expressions for the distance: cylinder, sphere, torus, cone, block. Expressions for these primitives can be found in [39].

- primitives with procedural distance evaluation: ellipsoid [40].

The distance to an ellipsoid [40] can only be computed by a numerical procedure, which requires solving a polynomial of degree 6. We extend the algorithm given in [40] to handle the numerically unstable cases, i.e. on the main planes of the ellipsoid.
**Complex primitives:** When complex, freeform, primitives are needed, some practical methods can be used. Given \( f \) an FRep for a complex shape, it can be for example a blobby object or a skeleton based object, a first method consists in polygonizing \( f \) (using algorithms described in [12, 71]) and then compute the distance to the polygonal mesh (using for example the algorithm described in Chapter 5, Section 5.1.4 or in [58]).

A second method consists in solving the following partial differential equation, called the restart equation [100, 90]:

\[
\frac{\partial d}{\partial t} = \text{sign}(f(p))(1 - |\nabla d|)
\]

(3.1.15)

with the initial condition

\[d(p, 0) = f(p)\]

where the function \( \text{sign} \) is defined by:

\[
\text{sign}(f) = \begin{cases} 
-1 & \text{if } f < 0 \\
0 & \text{if } f = 0 \\
1 & \text{if } f > 0
\end{cases}
\]

and \( t \) is an artificial time. Equation 3.1.15 is solved on a grid, the grid’s values can then be interpolated, using for example methods from [80], to obtain a function.

Another numerical method to obtain a signed distance function from a FRep \( f \) consists in searching for the closest point \( P = (x, y, z) \) on the implicit surface \( f = 0 \). This method computes also the values of the distance function on a grid, therefore the results may again need to be interpolated. Let \( P_0 = (x_0, y_0, z_0) \) be a point in space (a node on the grid), we search for the closest point \( P \) on the zero level-set of \( f \) – it means \( f(P) = 0 \) – and \( P - P_0 \) is parallel to the gradient \((f_x, f_y, f_z)\) of \( f \) at \( P \).

\[
R = \begin{pmatrix} 
\frac{f(P)}{f_x} \\
(y - y_0)f_x - (x - x_0)f_y \\
(z - z_0)f_x - (x - x_0)f_z
\end{pmatrix} = 0
\]

(3.1.16)
The system of equations 3.1.16 is solved by the damped Newton’s method for the column vector $\mathbf{P} = (x, y, z)$ with the initial solution $\mathbf{P}_0 = (x_0, y_0, z_0)$. We note $\mathbf{J}$ the Jacobian of the system $\mathbf{R}$ in Eq. 3.1.16. Equation 3.1.17:

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \alpha \mathbf{J}^{-1}(\mathbf{P}_k)\mathbf{R}(\mathbf{P}_k)$$ (3.1.17)

is iterated until $\mathbf{R}(\mathbf{P}_k)$ is small. $\mathbf{P}$ is set to $\mathbf{P}_k$ and the signed distance is given by:

$$d(\mathbf{P}) = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} \text{sign}(f(\mathbf{P}_0))$$ (3.1.18)

A last but no so accurate method is to use a normalization as described by Rvachev [84, 9]. Normalization of $f$ to the first order is obtained by:

$$f_1 = \frac{f}{\sqrt{f^2 + (\nabla f)^2}}$$ (3.1.19)

Then the normalization up to the order $n$ of $f$ is obtained by recurrence:

$$f_n = f_{n-1} - \frac{1}{n!} f_1^n \frac{\partial^n f_{n-1}}{\partial \nu^n}$$ (3.1.20)

where $\nu$ is the unit normal to the surface at the point of evaluation. The normalization of a function gives an approximation of the distance only close to the surface.

### 3.2 Automation of modeling by template FRep models

In the previous section, we saw how to build in a constructive way a solid defined by an approximation of the distance function. Modeling requires lots of skills and can be a difficult and time-consuming task. We introduce in this section the notion of template FRep to automate the modeling step [26, 25]. A practical case requiring automation consists in modeling existing objects acquired with scanners.

The FRep model refers here to the general case [70] and is not only limited to the distance-based model discussed in previous section. However the use of the distance or approximate distance function can provide better results in the fitting process of template models, see Chapter 5, Section 5.4.3 and [24].
3.2.1 Template FRep models

An FRep model can be built in a constructive way, with abstract parameters. The modification of these parameters can result in various different shapes, which can also be tuned to fit some modeling criteria. In the following, the notation $F(p, a)$ is used for a parameterized FRep, where $p = (x, y, z) \in \mathbb{R}^3$ is a point in the 3D space and $a = (a_1, \ldots, a_m) \in \mathbb{R}^m$ is a vector of $m$ parameters.

Template model can exist in specialized libraries for each application domain (mechanical design, human prosthesis design, and others) and may be reused, or need to be created by the user. In the latter case, a modeling work needs to be done by a designer. A parameterized model can be created using measurements or scans of a typical object. The model is required to keep basic ratios of the measured sample object and to proportionally change the dependent parameters according to introduced constraints. In case of scanned data available for a typical object, fitting of the template parameters can be also employed to establish basic ratios and constraints.

An example of a parameterized template FRep model, with different instances of the parameters, is illustrated in Fig. 3.17. The different model’s parameters represent the dimensions of the box, the diameters of the cylinders and their position.

![Figure 3.17: Illustration of a parameterized template FRep model](image)

Figure 3.17: Illustration of a parameterized template FRep model: a model is built with FRep in a constructive way with abstract parameters that can be tuned to satisfy some modeling criteria.
3.2.2 Fitting problem formulation

The problem is to recover a solid from a set of 3D points, $S = \{p_1, \ldots, p_N\}$, scattered on the surface of the object. Given $S$, the task is to find the best configuration for the set of parameters $a^* = (a_1^*, \ldots, a_m^*)$ so that the parameterized FRep model $F(p, a^*)$ closely fits the data points. $F(p, a)$ is an FRep model, made in a constructive way, which approximates the shape of the solid being reverse engineered. The vector of parameters $a$ control the final shape of the solid and the best fitted parameters should give the closest possible model according to the information provided by $S$.

For computing how close a given point is to the surface of the solid with the current set of parameters, a fitness function is needed. The FRep model $F(p, a)$ itself can serve for defining such a measure. Note that the better $F$ is an approximation of the Euclidean distance function, the more robust the fitting will be. The error of fit becomes the square of the defining function values at all points (the surface of the solid being the set of points with zero function value):

$$
error(a) = \frac{1}{2} \sum_{i=1}^{N} F^2(p_i, a)
$$

which can be also rewritten under the following form:

$$
error(a) = \frac{1}{2} \sum_{i=1}^{N} F_i^2(a) = \frac{1}{2} F'(a)F(a)
$$

where $F(.)$ is the vector with $F_i(.)$ as the $i-th$ component. Now, we are searching for the vector of parameters $a^*$ minimizing the error of fit from equation 3.2.1. We consider at first direct, also called local, methods for minimizing function with nonlinear parameters.

3.2.3 Nonlinear minimization of least squares by local methods

The best set of parameters $a^*$ is found by minimization of the least square error (equation 3.2.1). This least square error is usually a nonlinear function of the parameters $a$. Traditional methods for solving such problems are Levenberg-Marquardt
methods [75], [62] or Newton methods [22] (Full-Newton or Quasi-Newton). Such
Algorithms proceed iteratively from an initial set of parameters and try to converge
to a minimum in the parameter space. They are local methods in the sense that they
strongly depend on the initial parameters, which are used for starting the algorithm.

Such algorithms search in each iteration for a privileged direction to go in the
parameter space and for a step to move in that direction. Levenberg-Marquardt
and Newton type algorithms differ in the selection of direction and in the ways of
computing the step.

The function being minimized, in that case the least square error, is approximated
by a Taylor serie expansion to the second order:

\[
\text{error}(a) = F^t(a_k)F(a_k) + (a - a_k)^t J(a_k)^t F(a_k) + \frac{1}{2} (a - a_k)^t (J(a_k)^t J(a_k) + \sum_{i=1}^{N} F_i(a_k) \nabla^2 F_i(a_k))(a - a_k)
\] (3.2.3)

where \( a_k \) is the vector of parameters for the \( k \)-th iteration, \( J \) is the Jacobian of \( F \) and
\( \nabla^2 F_i \) is the Hessian of \( F_i \). Note that \( \nabla \text{error} = J^t F \) and \( \nabla^2 \text{error} = J^t J + \sum_{i=1}^{N} F_i \nabla^2 F_i \).

Equation 3.2.3 means that we consider that the least square error function can be
approximated locally by a quadratic model.

**Newton methods** compute the direction as the solution of the following equation:

\[
H (a_{k+1} - a_k) = -J^t F(a_k)
\]

where \( H \) is the Hessian matrix of the error function, or its numerical approximation
when it is not known analytically. Variant of Newton methods differs in whether they
use the exact Hessian or an approximation. Gauss-Newton for example uses \( J^t J \) as
an Hessian approximation.

**Levenberg-Marquardt** algorithm provides an efficient way for switching between
a Newton method and a steepest descent method for selecting a direction. This is
done by solving the following equation for the unknown $a_{k+1}$:

$$(J^TJ + \lambda I)(a_{k+1} - a_k) = -J^TF(a_k)$$

where $a_k$ is the current vector of parameter, $a_{k+1}$ is the next vector of parameter. $\lambda$ is adaptively changed during run-time to allow the use of a Newton method ($\lambda = 0$) or a steepest descent method (big value for $\lambda$).

These methods can in general guarantee only a convergence to a local minimum. For some parameter spaces with complex topology like, for example, where multiple local minima exist, such methods are likely to be trapped into a local optimum and to stop there. Good initial parameters are very important, because they will determine to which minimum the algorithm may converge. Usually, if the parameters are not in the neighborhood of the global minimum, it is unlikely to converge to it, and the method instead will reach a local minimum.

It is possible with some further analysis of the model to have some additional information for getting better estimation of the starting parameters. It may also be possible to restart the search algorithm with different starting points. Another method consists in using metaheuristics such as simulated annealing or genetic algorithm to perform the nonlinear optimization.

### 3.2.4 Simulated annealing

Simulated annealing has been proven to be one of the most effective method for solving combinatorial and continuous global optimization problems [60, 50, 75].

When trying to minimize an objective function usually only downhills are accepted, but within a simulated annealing algorithm some uphills may be accepted. This acceptance is made with a probability $p(T)$, which is initially close to 1, and then decreases to 0, when the temperature $T$ of the system reduces.

The procedure, governing the temperature evolution of the system, is called temperature schedule or cooling schedule.
Simulated annealing algorithm has been inspired by the behaviour of some thermodynamical process: in a liquid at high temperature, the molecules move freely with respect to one another; when the liquid is cooled down, the mobility of the molecules decrease, and finally stops. If the cooling is not too fast, then the system will finish in a state of minimum energy.

According to the Boltzmann law, a system in thermal equilibrium at temperature $T$ has its energy distributed probabilistically among all different energy states. Even for a low temperature there is a chance for the system to be in a high energy, so that it can escape the local minimum energy and finds a better one.

Convergence to an optimal solution can be theoretically guaranteed, but only after an infinite number of iterations controlled by the cooling schedule. In order to provide a finite time implementation, a proper cooling schedule is needed to simulate the asymptotic convergence behavior of the simulated annealing. For that reason, simulated annealing suffers from slow convergence and may wander around the optimum solution when high accuracy is needed.

A typical simulated annealing algorithm is given below:

**Simulated-Annealing**

1. Initialization. Choose an initial solution $x_0$. Fix the parameters for the cooling schedule: the initial temperature $T_{max}$, the epoch length $M$, the cooling reduction factor $\lambda$, the minimum temperature $T_{min}$.

2. The main iteration. Repeat $M$ times the following SA search. Generate a trial point randomly $x_{SA}$ within the feasible domain. Evaluate $f$ at the trial point $x_{SA}$ and accept the trial point $(x_{k+1} := x_{SA})$ if:
   (a) $\Delta f := f(x_{SA}) - f(x_k) < 0$ or
   (b) $\Delta f \geq 0$ and $p = \exp(-\Delta f / T) \geq r$ where $r$ is a random number in $(0, 1)$.

3. Termination. If the cooling schedule is completed ($T \leq T_{min}$) or the function values of two consecutive improvement trials become close
within a tolerance $\epsilon$ or when the number of total iterations exceeds some user threshold, then quit. Otherwise, decrease the temperature by setting $T := \lambda T$ and go to step 2.

3.2.5 Genetic algorithms

Genetic algorithms [42, 36] have been popular recently to solve problems of optimization (combinatorial or continuous). A genetic algorithm consists in a population of individuals, also called chromosomes, and operations, inspired by the mechanisms of natural selection and the law of genetics, which act on the elements of the population. An individual encodes a solution of the problem using an appropriate representation for this given problem. In the case of nonlinear optimization of real-valued functions, it consists in a set (an array) of parameters from the parameter space. The different operations acting on the individuals from a population are typically selection, crossover, and mutation.

Selection is the part of the reproduction process where individuals that will undergo mating are chosen. One individual has a chance to be selected proportional to its fitness. In the case of function minimization, the parameters with the lowest fitness value have the greatest chance to be selected. Some hazard is usually added to the selection process, so that the individuals with the worst fitness function have still a small probability to be selected.

Crossover is performed on the selected individuals. First the individuals are mated randomly, then each pair of individual to be mated may undergo a crossover as follow: let $l$ be the length of the individual, let $k$ be a random integer between 1 and $l - 1$, then the sub-parts of the individual between $k + 1$ and $l$ will be swapped. The idea of crossover is illustrated by Fig. 3.18.
Figure 3.18: Illustration of a crossover operation acting on two parents and producing two offsprings.
**Mutation** is an operation performed within an individual. It consists in a random alteration of one element in the individual with a given probability. Usually the probability of mutation is very low, especially compared to the one of crossover. However the mutation is an important part of the genetic algorithm, since it is responsible to the introduction of new genetic materials and diversity in the population. If the mutation rate in the genetic algorithm is too low, the genetic algorithm may converge prematurely to a uniform population, which may not contain the optimal solution. Mutation of an individual, represented by a string of characters, is illustrated by Fig. 3.19.

![Figure 3.19: Illustration of a mutation operation altering one gene.](image)

**3.2.6 Discussion**

Genetic algorithm and simulated annealing are good methods for finding a global optimum among many local optima. These methods are also particularly suitable for a parallel implementation, due to the relative independence of the evaluations of the objective function.

Practically it is useful to associate these global methods with local methods to refine the solution and/or accelerate the search. Combination of global and local
methods is explained in Chapter 4, Section 4.4.

Genetic algorithm and simulated annealing have both been successfully used in searching the parameters that best fit template FRep models to point-sets in the works [26, 25, 30].

It is possible to further automate the recovery process by automatically finding the FRep model from a segmented point-set. Once a model is obtained it is possible to parameterize it and further reuse the parameterized model.

3.3 Constructive tree recovery using a genetic algorithm

We propose an algorithm for recovering an FRep model defined by a constructive tree from a segmented point-set and a list of fitted primitives [27]. The goal is to further automate the modeling process by providing a way to recover constructive FRep models, that can also later be parameterized and used as valid template models.

The algorithm proposed in the following relies on the existence of a segmentation of the point-set and a list of primitives fitted to the segmented point-set. An example of a segmented point-set with the corresponding fitted primitives is illustrated in Fig. 3.20, where each fitted primitive is associated to one color.

The same algorithm can be applied to recover a FRep model from a boundary representation (BRep), since the BRep model naturally provides both the point-set and the primitives.

3.3.1 Description of the algorithm

Let us suppose that we have a set of points \( \{p_1, \ldots, p_n\} \) on or near the surface of the solid and a set of primitives \( \{f_1, \ldots, f_m\} \) fitted to the segmented point-set. Given a finite set of possible operations that can be applied to these primitives \( \{\lambda_1, \ldots, \lambda_l\} \), we are searching for an ordering of the primitives with operations acting on them.
such that the formula:

\[ f_{i_1} \lambda_{j_1} \cdots f_{i_m} \]  

(3.3.1)

is a correct FRep model for the solid defined by the point-set. In the above expression, \( j_k \in \{1, \ldots, l\} \), and the set \( \{i_1, \ldots, i_m\} \) is obtained from the set \( \{1, \ldots, m\} \) by a bijection and is used to order the primitives. A correct FRep model means that the defining function \( f = f_{i_1} \lambda_{j_1} \cdots f_{i_m} \) satisfies \( f > 0 \) inside the solid, \( f < 0 \) outside the solid and \( f = 0 \) on the boundary of the solid, defined by the discrete set of points.

If this formula is evaluated from left to right, it is clear that it corresponds to a tree structure (left unbalanced) with operations in the internal nodes and primitives in the leaves. Evaluation from left to right, using an intermediate variable temp, is
done as follow:

\[
\begin{align*}
temp & \leftarrow f_{i_1} \lambda_{j_1} f_{i_2} \\
temp & \leftarrow temp \lambda_{j_2} f_{i_3} \\
& \cdots \\
temp & \leftarrow temp \lambda_{j_{m-1}} f_{i_m}
\end{align*}
\] (3.3.2)

This representation, Eq. 3.3.1, comes from the fact that we need to encode each FRep in an individual to be processed by the genetic algorithm. This representation suits well the encoding, and is easy to evaluate.

The question is whether any constructive FRep can be encoded in that way. If the operations are only set-theoretic, then any FRep can be represented by a left unbalanced representation. Using DeMorgan transformations: \( X \setminus Y \rightarrow X \cap \overline{Y} \), \( X \cap Y \rightarrow X \cup \overline{Y} \), \( X \cup Y \rightarrow X \cap \overline{Y} \), and \( (\overline{X}) = X \), we transform the expression to an equivalent expression containing only \( \cup \) and \( \cap \). Then exploiting the fact that \( \cup \) and \( \cap \) are commutative: \( X \cup Y = Y \cup X \) and \( X \cap Y = Y \cap X \), we can switch internal nodes of the formula to obtain a left unbalanced representation.

Using blending operations [72], union or intersection, the representation of a constructive FRep by a left unbalanced representation is still possible when the blend is symmetric; if the blend is not symmetric, then the operations are not commutative, and a left unbalanced representation may not exist. Unary operations like space deformations [88]: rotations, scaling or any other inverse space mappings, do not pose any problems at all and can be appended to the primitives.

Note that, using DeMorgan laws to transform a constructive tree to a left unbalanced tree introduces the complement of point-sets. Practically it means that the primitives are oriented. When converting BRep models to FRep models it may generally be the case, since the BRep model requires a global and consistent orientation. When fitting primitives to a point-set, the orientation can be obtained from the orientation of the point-set, which is a difficult problem [40].
Because the points \( \{p_1, \ldots, p_n\} \) belong to the surface of the solid, a correct FRep should evaluate to 0 at each point. With the notation, \( f = f_{i_1} \lambda_{j_1} \ldots f_{i_m} \), it means \( \forall p_i, f(p_i) = 0 \). The problem can be reformulated as the search of a formula \( f \) such that \( \sum p_i f(p_i)^2 \) is minimum. A genetic algorithm is used for the minimization problem. This problem is combinatorial and the genetic algorithm is used to perform two tasks, which are 1) order the primitives and 2) find the correct binary operations to apply between two primitives.

An individual of the population represents a possible solution of the problem. In this case, it is a left unbalanced expression of a constructive FRep as illustrated by Eq. 3.3.1 (this is the phenotype or what exactly is the individual in the real world). Each individual contains \( m \) pairs of integers \( (op_k, L_k) \), \( 1 \leq k \leq m \), corresponding to the type of operations – \( op_k \) is an index to one of the operations from the set of possible operations – and \( L_k \) the position of the primitive \( k \) in the expression. The operation, indexed by \( op_k \), is applied between the primitive \( k \) and the preceding primitive, at the position \( L_k - 1 \), in the reconstructed expression. One pair is encoded by a bit string (a gene). The aggregate of the genes, encoding the pairs, describes the individual’s genotype. The expression encoded in an individual contains \( m \) operations, but a FRep expression using \( m \) primitives has in fact only \( m - 1 \) operations: one of the encoded operations is not used in the expression but is kept for the representation, i.e. appears in the genotype.

We give more details on the implementation as well as a practical example illustrating the algorithm in Chapter 4, Section 4.5.

### 3.3.2 Relation with genetic programming

Genetic programming evolves computer programs by a Darwinian process of evolution [52]. It is similar to a genetic algorithm, with the difference that the data structure used to encode the program (the solution of the problem) is a tree of potentially infinite size. The approach, that we used here, looks like genetic programming, but the size of the representation is of fixed length; consequently there are no advantages
in using genetic programming.

In fact, using genetic programming would result in long expressions with redundant information, as for example the sub-expressions: \( f \lor f \) (union of the solid with itself), \( f \land f \) (intersection of the solid with itself). Genetic programming can also lead to sub-expressions like \( f \setminus f \) which corresponds to the boundary only of the solid defined by \( f \) and is not suitable in solid modeling. However such an expression evaluates to 0 on the subset of points corresponding to the fitted primitive \( f \) and will not penalize the fitness function. Tracking such invalid sub-expressions or redundant information can only be done at the end, with algorithms for symbolic simplification.

### 3.3.3 Description of solids and primitives

A necessary condition, to be able to represent the solid by a constructive FRep, given in the form of Eq. 3.3.1, is that the list of primitives is sufficient to describe the solid. Shapiro and Vossler proved in the work [95] that boundary patches on the surface may not be enough to describe the same solid by CSG and that extra halfspaces may be needed to create a CSG representation of the same model. They describe how to detect such cases and how to construct these additional halfspaces.

We illustrate this notion in Fig. 3.21. The left part illustrates a boundary representation of a simple two-dimensional object, where each boundary patch has a different color. Building a constructive representation of this object, given the boundary patches, is however not possible. An additional halfspace is required as illustrated in Fig. 3.21, right.

The final CSG representation of the solid is obtained by taking the union of the disk with the additional halfspace, and then intersecting this solid with the remaining halfspaces:

\[
f = a \lor b \land c \land d \land e
\]

The algorithm proposed here does not automatically induce additional halfspaces. They need to be detected and constructed by the algorithm of Shapiro and Vossler [95] and inserted in the list of primitives. In some cases, however, circular or spherical
shapes are used to smoothly blend surfaces together; in these cases, additional halfspaces are not required by our algorithm, because the circular/spherical shapes are incorporated in the blending operations, which can be used as a valid operation within our algorithm. We also noticed that when the algorithm is applied to the recovery from point-set the need of extra halfspaces is dependant on the segmentation/fitting algorithm. In the example used above, if a rectangle is fitted instead of a collection of segments, then the extra halfplane is not needed because it is already embedded in the rectangle.
Chapter 4

Implementation and details of the algorithms and methods

4.1 The HyperFun project

The HyperFun project [1] is an open source system for FRep modeling. So far it has been used for experiments, research and teaching purposes. The system is based on FRep, which provides a mean for a high-level shape representation. The system HyperFun refers to the language specification for FRep based modeling but also to the set of available tools, which allow manipulating FRep objects. The system architecture of the HyperFun project is illustrated by Fig. 4.1.

The HyperFun system mainly consists in the following group of components: HyperFun models, the FRep library, interpreters/compilers, modelers, and applications.

4.1.1 HyperFun language

HyperFun [1] is a specialized language for modeling and description of functionally defined multidimensional geometric shapes. The language was kept as simple as possible by design but supports most of the FRep concepts.

An HyperFun model consists in one or more FRep objects, which are functions parameterized by input arrays of point coordinates \( x \) and numerical parameters \( a \). The numerical parameters are passed from outside the object; for example from another object, or from command line arguments. The function typically describes the
Figure 4.1: System architecture for the HyperFun project from [1]
geometry of the object; with the attributes extension of the FRep model [69], the
FRep function can also embed within its definition a set of scalar functions or calls
to these functions, which define the object’s attributes. The attributes’ values are
obtained from the HyperFun model through the optional s array. A function can be
constructed like in any typical programming languages by the help of assignment,
conditional and iterative statements. Expressions are composed using traditional
arithmetic and comparison operations. Mathematical functions like ‘sin’, ‘cos’, ‘tan’,
‘exp’, ‘sqrt’, and others are available in the language. Finally the set-theoretic op-
erations: intersection ‘&’, union ‘|’, difference ‘\’, negation ‘−’ and the Cartesian
product ‘@’ are keywords of the language.

For ease of use and development of complex geometric objects, various primitives
and operations have been implemented within the FRep library and are available
from the language. The FRep library can be easily extended by users to include
new primitives and operations for the language. Primitives available from the library
include common CSG primitives (block, sphere, cylinder, cone, torus), algebraic sur-
faces (ellipsoid, superellipsoid, elliptical cylinder and cone), skeleton objects, blobby
and soft objects, metaballs and others. The operations available from the library
include blending union and intersection, rotation, twisting, tapering, translation and
others.

An example of HyperFun model illustrating some of the points mentioned above is
given in Fig. 4.2. The model consists of three objects: one sphere and two cylinders.
Though the sphere primitive is available in the FRep library, we implemented it
in HyperFun language to illustrate the definition of auxiliary function and function
call. The two cylinders, ‘cylinder_x’ and ‘cylinder_y’, are defined using the cylinder
objects available from the library: ‘hfCylinderX’ and ‘hfCylinderY’. The model named
‘my_model’ is made from the union of the cylinders ‘|’ subtracted ‘\’ from the sphere.
Note the call to the previously defined function ‘my_sphere’.

Application softwares interact with HyperFun models through the interpreter,
implemented in C language, or through the compiler to Java bytecode.
4.1.2 HyperFun software tools

HyperFun software tools are freely available through the HyperFun web-site [45]. Let us briefly describe these softwares:

- **The HyperFun polygonizer**: This program polygonizes an HyperFun model, given through the form of a text file and interpreted by the interpreter, and displays the triangle mesh on the screen. It uses a simple command line interface, and allow the resulting triangle meshes to be exported to various formats such as: VRML, STL (for rapid prototyping), or POV-Ray meshes [74].

- **HyperFun for Windows**: This program is a programming environment for HyperFun, with a text editor, and various options available through menus and pop-ups. It is also based on the interpreter and can generate POV-Ray output and animation sequences.

- **HyperFun for POV-Ray** [74]: is a plugin to the version 3.1 of the popular raytracer POV-Ray. It allows using an HyperFun model as a normal POV-Ray
object. It is based also on the interpreter.

- HyperFun applet: is an applet to allow creation and visualization of HyperFun models through a browser [19, 31]. It is based on the HyperFun to Java bytecode compiler.

We discuss in the following the modifications of the HyperFun software tools to allow modeling with distance-based models as discussed in Chapter 3.

### 4.2 Web-based shape modeling using real distance functions

We have implemented primitives and operations for distance-based modeling within the HyperFun applet described in [19]. At first, we sum up the main points of the applet’s components. Then, we explain how the system is modified for distance-based shape modeling [31].

#### 4.2.1 HyperFun applet components

The applet system is mainly based on three components: a HyperFun to Java bytecode compiler, a polygonizer, and the applet providing the user interface. We briefly characterize the role of each component below.

**HyperFun to Java bytecode compiler**

The core of the modeling system is an HyperFun to Java bytecode compiler, its architecture is more largely described by R. Cartwright in his PhD dissertation [18], see also [19]. HyperFun code is translated to calls to the ByteCode Engineering Library (BCEL) API [101], resulting in generation of executable Java bytecode. In conjunction with the just-in-time compiler for Java bytecode, it allows for the fast evaluation of defining functions for shape models. The evaluation is needed when rendering the model by raytracing, polygonization, or other methods for rendering FRep objects.
Polygonizer in Java

Polygonization is the process that generates a polygonal approximation of an implicit surface (isosurface). The polygonization used in the applet is based on the algorithm described in [71]. This algorithm falls within a class called exhaustive enumeration as identified by [13]. There are two phases in the algorithm: spatial partitioning and cell polygonization. During spatial partitioning, the rectangular bounding box containing the object is divided into regular cubic cells. The polygonization of cubic cells has topological ambiguities on faces with four edge-surface intersection points. The algorithm in [71] and the one used here resolve the ambiguity by applying a trilinear interpolation in the cell, and a bilinear interpolation on the cell faces. The speed of the algorithm is improved here by using a look-up table similar to the one of the Marching Cubes (MC) algorithm [56]. The look-up table is generated using the algorithm of [71] and is used to resolve the topological ambiguities.

Applet interface

The interface of the applet shown Fig. 4.3 has four main parts: top left is the rendering window, bottom left is the error message text area, top right is the text area used to input the HyperFun model, and bottom right controls different options.

A model in HyperFun language is input in the dedicated area; pushing the “Polygonize” button compiles the HyperFun code to Java bytecode and calls the polygonizer to generate a polygonal mesh approximation of the object surface. Accuracy of the approximation is controlled by the number of subdivision steps of the bounding box along each axis and can be modified in the menu controls (bottom right). The definition of the bounding box may also be modified in the menus.

The rendering of the polygons is done in the rendering window using the Java3D API. The polygonal mesh can also be exported to a VRML file, to STL format for rapid prototyping, or to a mesh using the POV-Ray syntax for raytracing.
Figure 4.3: User interface of the HyperFun applet for distance-based shape modeling.
4.2.2 Extension for distance-based modeling

In order to support modeling with objects defined by Euclidean distance functions, the core and library of the compiler are modified. At first, the core compiler is modified to allow the use of the SARDF operations (union, intersection, and difference) instead of the min/max or R-functions. SARDF operations as described in Chapter 3, Section 3.1.2 are implemented in the Java language.

A menu, named “set-theoric”, is added to the applet to select the class of operations, i.e., SARDF for distance-based modeling, traditional R-functions, or min/max, and to select the radius threshold \( R \) for the SARDF operations, which controls the upper error bound in the distance approximation. A screenshot of the menu is given in Fig. 4.4.

![Figure 4.4: Menu for the selection of the set-theoretic operations.](image)

If “R-functions” (it includes also min/max) is selected, then the variable “alpha”
is used. It controls the smoothness of the R-functions. The value for “alpha” should be in \([-1,1]\), if it is outside this scope then the default value of 0.0 is used. Min/max can be obtained with alpha=1.0; If “SARDF” is selected, then the variable “radius” is used. It controls the error limit when smoothing the sharp corners (it corresponds to the radius of the biggest arc of circle used for smoothing the sharp corners). The value for “radius” should be in \([0, \text{Inf}]\); if it is outside this scope then the default value of 1.0 is used.

The HyperFun library is rewritten using primitives with distance functions and operations keeping the distance properties or at least an approximation. Available primitives and operations are:

- Primitives: hfSphere, hfEllipsoid, hfCylinderX, hfCylinderY, hfCylinderZ, hfTorusX, hfTorusY, hfTorusZ, hfBlock, hfQuadric, hfConeX, hfConeY, hfConeZ.

- Operations: hfShift3D, hfRotate3DX, hfRotate3DY, hfRotate3DZ, hfBlendIntSardf, hfBlendUniSardf.

Note that the list of primitives can be extended using the methods described in the previous chapter, Section 3.1.5.

Blending intersection and union (‘hfBlendIntSardf’ and ‘hfBlendUniSardf’) are implemented through a simple modification of the SARDF operations. The idea consists in using a constant circular arc approximation everywhere to replace every sharp corner in the contour lines of min/max; even the sharp corner at the origin is replaced now by a circular arc. In contrary to the SARDF union and intersection, there are no regions with a growing circular arc approximation. These two symmetric blending operations are entirely controlled by a single parameter \(R\), corresponding to the radius of the circular approximation. This parameter defines the shift to be applied to the line \(y = x\) in the \(\vec{x}\) and \(-\vec{x}\) directions, in order to define a bounding band that will enclose the circular arc approximation. We propose to illustrate the described blending operations here using an example of a vase, see Section 4.2.3, and more specifically one of its parts where blending union and blend on blend are used.
The area of interest of this shape is shown in Fig. 4.5 (a). This part is mainly composed of a horizontal block, a cone and some parts of a torus. Two blending union operations are applied in order to blend these three parts. In Fig. 4.5(b) and (c), we separate each blending function, i.e., respectively a blending union between the cone and the block and a blending union between the cone and the torus. In Fig. 4.5(d), the resulting object is shown where a blending union is applied between the two blended parts (blend on blend).

The possibility to export the mesh for the POV-Ray raytracer is added to the export menu (“Open/Save”) in order to do photo-realistic rendering. A new menu, as illustrated by Fig. 4.6, is also added to do cross-sections of the models. Cross-sections are done by planes perpendicular to x, y, and z (it is straightforward to extend it to any vector). A plane is defined by choosing one of the “Axis” and setting the corresponding “Value”. For example, choosing “x” and “0.0” as in Fig. 4.6, makes a cross-section of the model by the plane \( x = 0.0 \). The fields “Lower” and “Upper” indicates the values of the bounding box in the selected plane. Finally, the user selects an image size by giving its “width” and “height”.

Figure 4.5: Blending operations: (a) Initial object composed as a vertical cone, a torus-like shape and a block shape. (b) Blending between the block and the cone. (c) Blending between the torus and the cone. (d) Blending between the block and the cone, and blending between the previous resulting object with the torus.
4.2.3 Examples of distance-based modeling with the applet

After modeling objects using the extended HyperFun applet, one can export the resulting object either directly as a HyperFun text file, as an STL file for rapid prototyping, or as a polygonal mesh, using the POV-Ray syntax, for further photorealistic rendering. We propose herein to illustrate this feature with two examples, one related to artistic modeling and rendering, and another to mechanical Computer-Aided Design.

"CyberVase"

The first example related to artistic modeling is a vase model. Figure 4.7 shows the polygonal model of the object, modeled and rendered using the HyperFun applet. This object is modeled using the following primitives: ellipsoids, tori, cones and blocks. For the operations, SARDF unions, SARDF intersections, and SARDF blending are used. As one can see, complex objects can be modeled through the applet interface. Although this object could be rendered directly as a legal isosurface object in the raytracing engine POV-Ray (with adequate library extension), we
Figure 4.7: Polygonal model of a complex artistic vase modeled using the Java Hyper-Fun applet and distance functions and rendered within the applet (Model by Benjamin Schmitt).
choose to export this object as a polygonal mesh in order to obtain a photo-realistic rendering of this object. Let us emphasize here the importance of using implicit objects for web-based modeling and as a format for exchange of shapes. The HyperFun file size is less than 5Kb, whereas the polygonal export produces a file larger than 10 Mb (compressed). This large size for the polygonal model is due to, among other factors, the smooth blend of the vase shape that requires a large amount of triangles to approximate properly the original shape. Figure 4.8 shows the vase rendered with POV-Ray using transparent material. The vase example illustrates that non-trivial objects can be modeled and that high-quality, photo-realistic rendering can be done on the resulting objects. Note that exporting to a mesh for rendering by raytracing may be faster than raytracing the implicit surface directly. Raytracing a mesh involves finding the intersection between a ray and a triangle, which is faster to compute than solving the nonlinear equation \( f = 0 \) where \( f \) is the FRep model.

**CAD object**

The second example, illustrated by Fig. 4.9, is related to Computer-Aided Design. In this example, a CAD mechanical part is modeled using the HyperFun applet. The constructive approach and the distance-based modeling allow one to meet the requirements imposed on this shape, i.e., locations of the holes, thickness of the blocks at the end of each branch, and others. The most intriguing result of the distance-based modeling approach is the distance field of the object, illustrated in Fig. 4.10 by a cross-section by the plane \( z = 0 \). The cross-section is created using the 'cross-section' menu of the java HyperFun applet. Each primitive of this object is defined by Euclidean distance, and the set-theoretic operations are defined using SARDF operations. Therefore, the distance field of the object gives an approximation to the exact Euclidean distance function and can be used in further applications such as heterogeneous object modeling [10].
Figure 4.8: High quality photo-realistic rendering of the vase modeled using the distance-based HyperFun applet, and rendered using the POV-Ray raytracer.
Figure 4.9: Raytraced mechanical CAD object modeled using our applet and distance functions. The original STL file for the CAD object is courtesy of Elena Kartasheva.
Figure 4.10: Distance field of the CAD object at the cross-section by the plane $z = 0$. SARDF operations are defined with a radius of 0.5.
4.3 HyperFun models with approximate distance-based functions

To allow creation of HyperFun models with approximate distance-based functions, the interpreter, written in C language, is also modified. We provide some basic informations on the HyperFun interpreter, then explain the modifications done to allow SARDF modeling.

The interpreter of HyperFun is used in several software tools such as: the HyperFun polygonizer, the HyperFun POV-Ray plugin, and HyperFun for Windows.

4.3.1 The HyperFun interpreter

It is written entirely in C language. Application softwares deal with the interpreter by the mean of two functions: ‘parse’ and ‘calc’.

‘parse’ reads an HyperFun model given as a text file. It checks the syntax of the input HyperFun model and to some extent the semantic. It builds an internal representation that will be interpreted by the ‘calc’ function.

‘calc’ takes as input the internal representation generated by ‘parse’ and evaluates it. The result is the value of the HyperFun model at every points of evaluation. Error checking is performed both during ‘calc’ and ‘parse’ and indicates the type of error and location in the HyperFun script.

The interpreter deals with the HyperFun library through a structure containing pointers to functions corresponding to the implementations of the library’s primitives and operations. Extending the HyperFun library requires implementing the functions and adding the corresponding pointers to the structure.

4.3.2 SARDF extension

We have implemented SARDF operations as described in Chapter 3, Section 3.1.3 in C language. The different approaches in quadrant 3 for the intersection and quadrant 1 for the union are implemented and the user can select the approach to be used. Each
approach is implemented in a different function and the selection is performed in the
HyperFun model definition through functions from the HyperFun library.

The function to compute the solution of the polynomial of degree four required for
the SARDF operations is also implemented in C in a different module and called by the
SARDF operations. Solutions to polynomial of degree four are known algebraically.

Even if the implementation of SARDF operations is not really complicated, it is
longer than the simple min/max or the R-functions. For that reason and for sim-
plicity of management of the code of the interpreter, the implementation of SARDF
operations are kept in auxiliary functions and not inside the code of the interpreter
like min/max and R-functions.

The HyperFun interpreter is used by several software tools, and is not necessarily
embedded in a graphical user interface. For that reason, the switch between SARDF
operations, R-functions and min/max can not rely on informations given to the in-
terpreter through pop-ups or selection menus. The best solution is to add functions
in the API of the HyperFun library to select the different options such as: use of
R-functions, SARDF or min/max, value of the alpha parameter for the R-functions,
or value of the radius for the SARDF operations. By default, the interpreter uses
R-functions for the set-theoretic operations. When doing so, we keep an ascendant
compatibility: HyperFun models for previous versions of the interpreter, can still be
interpreted correctly.

An example of HyperFun model allowing the use of SARDF operations is illus-
trated by Fig. 4.11.

Note the use of the functions from the library: 'hfUseSardf()' to set the set-
theoretic implementation’s mode to SARDF, 'hfSetRadius' to set the value of the
variable radius used by the SARDF operations, and 'hfUseRFunctions()' to set the
set-theoretic implementation’s mode back to R-functions. It is possible to mix within
a single HyperFun model the use of SARDF and R-functions in contrary to the applet
version.
-- main HyperFun model
-- allow for SARDF operations and modeling
my_model(a[3], a[4]) {
    array center[3],
    center = [0, 0, 0];

    -- use SARDF operations for set-theoretic operations
    tmp = hUseSARDF();
    -- set the radius in quadrant 1 to 1
    tmp = hSARDFSetRadiusQuadrant1(1);
    -- and to 1 in quadrant 3
    tmp = hSARDFSetRadiusQuadrant3(1);

    sphere = hDSphere(x, center, 5);
    cylinder_x = hDCylinderX(x, center, 2);
    cylinder_y = hDCylinderY(x, center, 2);

    -- use SARDF for the union
    cyl_union = cylinder_x | cylinder_y;

    -- change to R-functions
    tmp = hUseRFunctions();

    -- uses R-functions for the subtraction between sphere and cyl_union
    my_model = sphere \ cyl_union;
}

Figure 4.11: Example of an HyperFun model using SARDF operations, R-functions and distance-based primitives
The primitives defined by the Euclidean distance function are added to the HyperFun library. In order to distinguish them from the “normal” primitives, the letter ‘D’, for distance, is inserted in the name of the primitive after the prefix ‘hf’. See for example the listing in Fig. 4.11, which has calls to ‘hfDSphere’ or ‘hfDCylinder’.

Besides the SARDF set-theoretic operations, the HyperFun library of the interpreter is extended with the following primitives, operations and utilities:

- **Primitives:** hfDSphere, hfDEllipsoid, hfDCylinderX, hfDCylinderY, hfDCylinderZ, hfDTorusX, hfDTorusY, hfDTorusZ, hfdBlock, hfdQuadric, hfdConeX, hfdConeY, hfdConeZ.

- **Operations:** hfSardfInt1, hfSardfInt2, hfSardfIntG1, hfSardfIntG2, hfSardfUnion1, hfSardfUnion2, hfSardfUnionG1, hfSardfUnionG2, hfSardfSub1, hfSardfSub2, hfSardfSubG1, hfSardfSubG2, hfbIntSardf, hfbUniSardf.

- **Utilities:** hfSardfSetRadiusQuadrant1, hfSardfSetRadiusQuadrant3, hfSardfGetRadiusQuadrant1, hfSardfGetRadiusQuadrant3, hfSardfSetAngle, hfSardfGetAngle, hfuMinMax, hfuRF, hfuSardf.

Contrary to the Java applet, it is possible to use within one HyperFun model, primitives and operations defined using an approximate Euclidean distance function or not. The script in Fig. 4.11 contains for example set-theoretic operations implemented by SARDF and R-functions; the primitives are all defined by the distance function. In the Java version, the choice through the GUI of SARDF or R-functions selects for the current modeling session the use of only SARDF operations and distance functions or R-functions and algebraic distance functions.

The functions ‘hfSardfInt1’, ‘hfSardfUnion1’ and ‘hfSardfSub1’ implement the first approach for SARDF operations respectively in the quadrants 3, 1 and 3 as described in Chapter 3, Section 3.1.3, which consists in bounding the growth of the radius. The functions ‘hfSardfInt2’, ‘hfSardfUnion2’ and ‘hfSardfSub2’ implement the second approach, consisting in a growing radius, bounded by two lines, opened by a fixed angle.
The functions suffixed by 'G' (before the number '1' or '2' indicating the approach used) also compute the gradient of the resulting function. Computing the gradient is done using Equation 3.1.14. It requires the partial derivatives of the FRep arguments of the SARDF operations as well as the partial derivatives of the SARDF operations. Partial derivatives of the SARDF operations are easily obtained using calculus; the partial derivatives for SARDF intersections are given for example by equations 3.1.8, 3.1.9, 3.1.10, 3.1.11, 3.1.12, and 3.1.13.

4.4 Fitting template FRep model

Fitting a template FRep model to a point-set is done by global search methods based on metaheuristics such as simulated annealing [26, 25] or genetic algorithm [30]. Practically, the simulated annealing or the genetic algorithm is combined with a local search method, such as a Newton based method or Levenberg-Marquardt to accelerate the search and enhance the accuracy.

4.4.1 Implementation of an hybrid method

We have implemented the simulated annealing and direct search methods, such as Levenberg-Marquardt or quasi-Newton type, in C. The two algorithms, simulated annealing and the direct search method, are combined in a two-step process. The first step (simulated annealing) should give a configuration in the parameter space being in the vicinity of the global minimum and thus should help avoiding local minima. The second step (the local method) should guarantee a faster convergence by avoiding that the algorithm wanders around the solution and should also improve the quality of the fitted parameters.

At the beginning of the fitting process, the input consists of a preliminary selected or specially built parameterized FRep model, a set of 3D points, and an initial parameters’ estimation. The values of the initial parameters are not so crucial, because the sampling algorithm, simulated annealing, can escape local minima. When the parameters have geometric interpretation, it is of course possible to guess initial
values, even if not accurate at all. In the case when parameters have less obvious meaning, like the coefficients in blending operations, it is more difficult to provide good initial estimation, therefore, some random initial values in the parameter space may be chosen.

In order to avoid complete blind search in the simulated annealing algorithm, a point is generated in the neighborhood of the current evaluation point. Then the next exploration point is generated in the direction where the function decreases.

Switching from simulated annealing to the local method is done when: a given number of iterations (of simulated annealing) is done, or the cooling schedule is completed, or the function values of two consecutive improvement trials become close within a given tolerance. If the switch from simulated annealing to the direct search method is done because a number of iterations of simulated annealing has been done, then simulated annealing will be run again (continued) after the direct method is completed, starting with the best fitted parameters obtained from the direct method; these two steps – simulated annealing and the direct method – are iterated a given number of times. In the other cases, i.e. the cooling schedule is completed or the function values of two consecutive trials are close, the algorithm terminates after the direct search.

Additionally, a visual feedback is proposed to the user, such that if the shape corresponding to the actual parameters looks close to the point-set, the user can switch to the direct method.

The goal of the two-step method described above is to provide a fitting system for parameterized FRep models that does not require a good initial configuration, can escape local minima, and has an acceptable convergence rate.

4.4.2 Using a genetic algorithm for the global search

A different implementation can use a genetic algorithm instead of simulated annealing [30]. The use of a genetic algorithm requires the determination of some fundamental issues: the genotype, the initial population, the fitness function, the operations on
the genotypes, and the termination criterion. Each of these issues is described below:

1. **genotype**: the parameters of the model to estimate (the vector \( \mathbf{a} \) in Eq. 3.2.1); they are encoded as a vector of floating point values following [61],

2. **initial population**: a set of random individuals chosen within the bounded search domain,

3. **fitness function**: the function that quantifies the fitness of an individual to the environment by considering its genotype. In our case, it corresponds to the function defined by the equation (3.2.1),

4. **operations on genotypes**: they define alterations on genotypes in order to make the population evolve during generations. Three types of operations are used:

   • **individual mutation**: genes of an individual are modified in order to better adapt to the environment. We use the Non-Uniform mutation process which randomly selects one chromosome \( x_i \), and sets it equal to a non-uniform random number:

     \[
     x'_i = \begin{cases} 
     x_i + (b_i - x_i)f(G) & \text{if } r_1 < 0.5 \\
     x_i - (x_i + a_i)f(G) & \text{if } r_1 \geq 0.5
     \end{cases} \quad (4.4.1)
     \]

     where
     \[
     f(G) = \left( r_2 \left(1 - \frac{G}{G_{\text{max}}} \right) \right)^b
     \]
     \( r_1, r_2 \): uniform numbers in the interval [0,1]
     \( a_i, b_i \): lower and upper bound of chromosome \( x_i \)
     \( G \): the current generation
     \( G_{\text{max}} \): the maximum number of generations
     \( b \): a shape parameter

   • **selection of an individual**: individuals that are not adapted to the environment do not outlive to the next generation. We used the normalized geometric ranking selection method which define a probability \( P_i \) for each
individual $i$ to be selected:

$$P_i = \frac{q(1-q)^{r-1}}{1 - (1-q)^n}$$

(4.4.2)

where

$q : $ the probability of selecting the best individual

$r : $ the rank of individual, where 1 is the best

$n : $ the population size

- **crossing-over**: two individuals can reproduce by combining their genes. We use the arithmetic crossover which produces two complementary linear combinations of the parents:

$$X' = aX + (1-a)Y$$

$$Y' = (1-a)X + aY$$

(4.4.3)

where

$X, Y : $ genotype of parents

$a : $ a uniform number in the interval $[0,1]$

$X', Y' : $ genotype of the linear combinations of the parents

5. **stopping criterion**: We choose to consider the stability of the standard deviation of the evaluation criterion of the population.

Following [61], we use a real-valued encoded genotype, with operations on genotype chosen in consequence to work on real-valued encoded genotypes. Usually, a real-valued GA is an order of magnitude more efficient than binary GA in term of CPU, and offers a higher precision.

The first prototype implementation [30] was written in Matlab and later rewritten in C++ using the GALib library written by Matthew Wall [109].

### 4.4.3 FRep evaluation for the fitting algorithm

Simulated annealing, the genetic algorithm and the local search method – quasi-Newton or Levenberg-Marquardt – require the evaluation of the FRep for a given
vector of parameters. The local methods require also the evaluation of the partial
derivatives with respect to the different parameters.

Partial derivatives are computed by finite differences. For example, the partial
derivative of a function $f$ of variables $x, y, z$ with respect to $x$ is computed by:
\[
\frac{\partial f}{\partial x} = \frac{f(x+h,y,z) - f(x-h,y,z)}{2h}
\]
with $h$ being a small floating point value, 0.0001 in our implementation.

The function evaluation is viewed as a “black-box” function, that takes as input
two arrays of floating point values and returns a floating point value. The first array
contains the current point coordinates and is obtained from the point-set. The second
array contains the current vector of parameters computed by the fitting algorithm.

One possible implementation for the function evaluation “black-box” is to use
the HyperFun interpreter. A template FRep model is then defined by a normal
HyperFun file, the array $x$ of the HyperFun model contains the point coordinates and
the array $a$ contains the values of the parameters. The elements of the array $a[1]
\ldots a[n]$ are used inside the HyperFun model to parameterize the model. From a user
point of view, this approach is the most convenient. However, it is not so efficient
in term of speed, because the FRep model is interpreted. If speed is an issue, the
user can always implement the parameterized FRep model in C or fortran language.
Examples of computing times for fitting different objects implemented in C are given
in Chapter 5, Section 5.4. The drawback of this approach is that experimenting with
new parameterized FRep models requires recompiling the entire application.

It has been shown that Euclidean fitting is better than algebraic fitting in terms
of accuracy and robustness [24, 57]. Euclidean fitting means that the shape to be
fitted is defined by the Euclidean distance function to the boundary of the shape.
Following this observation, we use when possible SARDF operations and distance-
based primitives. The example in Chapter 5, Section 5.4.3, illustrates this point.
4.5 Implementation’s details of the constructive tree recovery algorithm

We give details on the implementation of the approach described in Chapter 3, Section 3.3 for recovering a constructive FRep model using a genetic algorithm.

The genetic algorithm is implemented in C++ language using the GALib library [109]. A simple genetic algorithm, as introduced and discussed by Goldberg [36], is used. This genetic algorithm uses non-overlapping populations. For each generation, a new population is created by selecting from the previous population, and mating to produce the offsprings for the new population. These two steps are repeated until the termination criterion is met.

The fitness function is defined by:

\[ \phi(g) = \sum_{p_i \in S} F_g(p_i)^2 \quad (4.5.1) \]

where \( g \) is an individual in the current population; \( S \) is the point-set, with points on or near the surface of the solid; \( F_g \) is the FRep encoded by the individual \( g \) and corresponds to a left unbalanced tree (Eq. 3.3.1).

4.5.1 The genotype, and its decoding (finding an object’s phenotype from its genotype)

We use a one dimensional binary string (‘GA1DBinaryString’) as the data structure to encode an individual’s genotype in the genetic algorithm. As explained in Chapter 3, Section 3.3, an individual corresponds to an array of \( m \) pairs \( (op_k, L_k) \), \( 1 \leq k \leq m \), where \( op_k \) is an index to one of the operations, and \( L_k \) is the priority level – or position – of the primitive \( k \) in the expression. Each pair is encoded as a sequence of bits (a gene).

We use an example to explain in more details how the decoding and encoding of the individuals are performed; this example consists of 16 primitives and the possible operations are union, intersection and blending intersection. Each pair \( (op_k, L_k) \) can
then be encoded by a gene of size \(2 + 4 = 6\) bits. An operation \(\text{op}_k\) can be encoded in 2 bits, since three operations are considered. The last 4 bits are used to encode the index of the \(k^{th}\) primitive (priority level) \(L_k\) in the expression construction. An expression (phenotype) is thus encoded by a string of size \(6 \times 16 = 96\) bits, i.e. an array of 16 genes of size 6 bits.

For a given binary string, we need also to be able to reconstruct the expression to evaluate it, i.e. finding the phenotype of an individual from its genotype. First, each gene, corresponding to a string of bits, is decoded to a pair of integers \((\text{op}_k, L_k)\). Then we use the following procedure:

- select the next pair \((\text{op}_k, L_k)\) with the lower \(L_k\) value
- in case of several pair with same \(L_k\) value, take the one with the minimal index \(k\) in the initial list

This procedure sorts the pairs within the array with respect of the priority level \(L_k\) of the primitive in the expression. It orders the primitives within the expression.

This procedure is illustrated in the following example. Let a string be encoded by the following 16 pairs \((\text{op}_k, L_k)\). Where \(\text{op}_k \in \{0, 1, 2\}\), and \(L_k \in \{0, \ldots, 15\}\).

\[
\begin{align*}
    k = 1 : (0, 15), & \quad k = 2 : (0, 1), & \quad k = 3 : (1, 0), & \quad k = 4 : (1, 2), & \quad k = 5 : (1, 1), & \quad k = 6 : (1, 0), \\
    \ldots, & \quad k = 16 : (1, 7).
\end{align*}
\]

Following the procedure described above, we select the pairs in the following order: 3, 6, 2, 5, 4 \ldots 1 and then the reconstructed expression becomes: \(F = (\text{op}_3)f_3\text{op}_6f_6\text{op}_2f_2\text{op}_5f_5\text{op}_4f_4\ldots\text{op}_1f_1\). The first operation is in parenthesis because it is not taken into account in the evaluation and is here only for the symmetry of the expression. Finally each operation can be replaced. Let us suppose that \(\text{op}_i == 0\) corresponds to union (||), \(\text{op}_i == 1\) for intersection (&), and \(\text{op}_i == 2\) for blending intersection (&&); we get the following expression: \(F = f_3\&f_6|f_2\&f_5\ldots|f_1\). This expression is evaluated from left to right, as explained in Chapter 3, Section 3.3, Eq. 3.3.2. The evaluation of this expression at each point of the point-set serves to define the fitness function for a given individual (Eq. 4.5.1).

We need to define now operations to operate on the individuals.
4.5.2 The genetic operations

There are at least three operations that need to be defined for a genetic algorithm:

- the selection (selection of individuals in the previous population),
- the crossover (mating selected individuals to create new offspring),
- the mutation (mutate the selected individual)

We discuss in the following the algorithms used for their implementations.

Selection:

The selection is implemented by the roulette wheel algorithm. The idea of the roulette wheel is to choose randomly between all the individuals of the current population, with a higher probability to select an individual with a good fitness value.

Mutation:

The first algorithm used for the mutation is a random flip of bits. If an individual is selected for a mutation, then 1 bit of the individual is chosen randomly and flipped (meaning that if the value of the chosen bit were 1 then it would become 0 and 0 would become 1). This algorithm is the default one. It is however not aware of the problem representation.

We have also implemented a different algorithm taking into account the structure of the problem. Let us suppose, that there are $m$ pairs, and that a pair is coded using $p$ bits as a gene. Note that with the example above $m = 16$ and $p = 6$. If an individual is selected for a mutation, then the mutation is done as follow:

1. choose randomly a position in the string
2. find the beginning of the $p$-bit sequence it belongs to
3. flip all the $p$ bits of the sequence
Crossover:

The default crossover operation is a one point crossover. Two individuals are selected from the previous population to be mated, then one point is chosen randomly in both individuals and the different parts are swapped.

By default, it is not aware of the structure of the problem, therefore the following crossover can be used:

1. choose randomly a position in the string of parent 1
2. find the beginning of the $p$-bit sequence it belongs to
3. find the symmetric position in the parent 2
4. swap the subparts of the parents to generate the two children
Chapter 5

Experimental results and applications

5.1 Constructive modeling with SARDF: quality of approximation

In this section, we look at the quality of the approximation of the Euclidean distance function obtained when using the SARDF framework, consisting in SARDF operations and primitives defined by the Euclidean distance function.

5.1.1 Comparison of different implementations for intersection

We look in the following at different functions implementing the set-theoretic intersection operation: min, R-function, and SARDF. The R-function used is \((x, y) \rightarrow x + y - \sqrt{x^2 + y^2}\). The SARDF intersection is as described in Chapter 3, Section 3.1.3 with a radius of 0.2 in the quadrant 1 and an angle of 0.0 in the quadrant 3 (the angle approach is used). For each function, the contour plot and the extremal values on a Cartesian grid: \([-10, 10] \times [-10, 10]\) are given. It is important to notice that the point-set: \(\{(x, y) : \text{intersection}(x, y) = 0\}\) corresponds geometrically to an L shape, with infinite segments. So the distance field of interest (the function value) is the distance to this L shape.
The Figure 5.1 illustrates the contour plots of the intersection implemented by min (top left), the R-function \((x, y) \rightarrow x + y - \sqrt{x^2 + y^2}\) (top right), and the SARDF intersection (bottom).

Figure 5.1: Contour plots of different implementations of intersection: min (top left), R-function (top right), SARDF intersection (bottom)

The min function (Fig. 5.1, top left) is not differentiable at any point of the line \(y = x\), as it can be seen by the sharp angle on every contour line; it represents however a reasonable approximation of the distance function, and is often used in constructive modeling with Euclidean distance fields (see [58], page 15). Notice that min is failing to represent truly the distance in the third quadrant, with both \(x < 0\) and \(y < 0\): the contour lines should be circular arcs centered in the origin. Extremal values of min on a \([-10, 10] \times [-10, 10]\) Cartesian grid are respectively \(-10\) and \(10\).

The R-function \((x, y) \rightarrow x + y - \sqrt{x^2 + y^2}\) (Fig. 5.1, top right) is smoother than min and at least \(C^1\) except at the origin. For that reason it has interesting properties in engineering [10]. It is however a poor approximation of the distance function; the
extremal values are respectively of $-34.1421$ and $5.85786$, underlining the explosion effect of the resulting function mentioned earlier.

The SARDF intersection (Fig. 5.1, bottom) is a good approximation for the distance function; contrary to min, it is smoother. This function is in $C^1$ except the origin. Note that in the upper quadrant $x > 0$ and $y > 0$ the function has a very small radius for the circular arc approximation, giving the impression that it has sharp corners similar to min, but it is $C^1$ contrary to min. The extremal values are respectively $-14.1421$ and $9.9415$. The parameters of the SARDF function are the radius of 0.2 and the angle of 0.0, note that we use the second approach for the third quadrant. In that configuration, the function corresponds to the exact distance in the third quadrant to the L shape. In the general case, when applied to two given functions, the distance value for the overall function is not exactly kept however; the circular approximation in the functional space is transformed to an ellipse in the geometrical space, whereas the distance to a point resulting from the intersection of lines should be circular in the geometrical space.

5.1.2 Comparison of the gradients’ norms for different implementations of the intersection

We saw that the Euclidean distance function is the vanishing viscosity solution of the Eikonal equation, Chapter 2, Section 2.1, Eq. 2.2.1. Considering the function $(x, y) \rightarrow \text{intersection}(x, y)$ as an approximation of the Euclidean distance to an L shape, we study the evolution of the norm of the gradient of the function $\text{intersection}$, when the gradient exists.

This study gives some indication on the behaviour of the norm of the gradient for the function $\text{intersection}$. A more general study is however impossible, because in practical modeling, the function $\text{intersection}$ is applied to functions, representing the distance fields of the arguments, and the gradient in a Cartesian coordinates system takes into account the gradients of the arguments, see Eq. 3.1.14 in Section 3.1.5, Chapter 3 or Eq. 5.1.1 and 5.1.2 in the following.
Let \( u(x, y, z) \) be the resulting function, constructed by applying a set-theoretic operation \( F \), on two functions \( h_1(x, y, z) \) and \( h_2(x, y, z) \). The norm of the gradient in Cartesian coordinate of \( u \) is given by:

\[
|\nabla u|^2 = \left( \frac{\partial F}{\partial h_1} \right)^2 |\nabla h_1|^2 + 2\left( \frac{\partial F}{\partial h_1} \right)\left( \frac{\partial F}{\partial h_2} \right)(\nabla h_1 \cdot \nabla h_2) + \left( \frac{\partial F}{\partial h_2} \right)^2 |\nabla h_2|^2 \quad (5.1.1)
\]

which introduces the gradients of the arguments \( h_1 \) and \( h_2 \). If \( h_1 \) and \( h_2 \) are distance functions, they are solution of the Eikonal equation 2.2.1. Equation 5.1.1 simplifies to:

\[
|\nabla u|^2 = \left( \frac{\partial F}{\partial h_1} \right)^2 + 2\left( \frac{\partial F}{\partial h_1} \right)\left( \frac{\partial F}{\partial h_2} \right)(\nabla h_1 \cdot \nabla h_2) + \left( \frac{\partial F}{\partial h_2} \right)^2 \quad (5.1.2)
\]

In the two-dimensional Euclidean space, taking \( h_1(x, y) = x \) and \( h_2(x, y) = y \), Eq. 5.1.2 simplifies to:

\[
|\nabla u|^2 = \left( \frac{\partial F}{\partial x} \right)^2 + \left( \frac{\partial F}{\partial y} \right)^2 \quad (5.1.3)
\]

which explains why looking at the behaviour of the norm of the gradient of the intersection function gives some indication on the general behaviour of the gradient’s norm. In the general case, when the two arguments are not orthogonal, an additional term \( 2\left( \frac{\partial F}{\partial h_1} \right)\left( \frac{\partial F}{\partial h_2} \right)(\nabla h_1 \cdot \nabla h_2) \) should be taken into account.

**min/max:** \((x, y) \rightarrow \text{min}(x, y)\) has no defined gradient on the line \( x = y \). Everywhere else, the gradient has a unit norm.

**R-function:** Figure 5.2 is a contour plot of the gradient’s norm of the R-function \((x, y) \rightarrow x + y - \sqrt{x^2 + y^2}\). The plot is over the domain \([-10, 0] \cup [0, 10] \times [-10, 0] \cup [0, 10]\). The R-function has no defined gradient in \((0, 0)\). The values range from 0.414214 to 2.41421. The gradient has a unit norm only when one of its arguments is 0. Consequently an object made by applying R-functions to primitives has a unit norm on the surface of the solid. Everywhere else it quickly increases or decreases.
Figure 5.2: Contour plot of the gradient’s norm of the R-function \((x, y) \rightarrow x + y - \sqrt{x^2 + y^2}\). The values of the norm range from 0.414214 to 2.41421.

**SARDF operation:** Figures 5.3 and 5.4 contain contour plots of the gradient’s norm of the SARDF intersection for different parameters. The different possible parameters include: the values of the radius in the quadrants 1 and 3, if the first approach is used, or the value of the radius in the quadrant 1 and the value of the opening angle in quadrant 3, if the second approach is used.

Figure 5.3 illustrates contour plots of the gradient’s norm of the SARDF intersection, when the bounded approach is used in the third quadrant (first approach). Figure 5.3, top left corresponds to radius of 0.1 and 0.1; Fig. 5.3, top right corresponds to radius of 0.5 and 0.5. And finally, Fig. 5.3, bottom corresponds to radius of 5.0 and 5.0. Table 5.1 contains some values of the gradient’s norm at different points and for different values of the parameters. The norm of the gradient is equal to 1 everywhere, but in the band where the circular approximation is performed. In this area, the value of the gradient’s norm is below 1. When increasing the maximum allowed radius for the approximation, the norm of the gradient quickly decreases away from 1. From Fig. 5.3, bottom, it appears also that the gradient’s norm is lower in the first quadrant than in the third quadrant. It appears that a low value for the maximum allowed radius is better for distance approximation purpose.

Figure 5.4 illustrates contour plots of the gradient’s norm of the SARDF intersection, when the angular approach is used in the third quadrant (second approach).
Figure 5.3: Contour plots of the gradient’s norm of the SARDF intersection, when the bounded radius approach (first approach) is used in the third quadrant. Values of the radius are respectively: top left: 0.1, 0.1; top right: 0.5, 0.5; bottom: 5.0, 5.0

Table 5.1: Value of the norm of the SARDF gradient for different values of the parameters, when the first approach in the third quadrant for constructing the SARDF intersection is used.
Figure 5.4: Contour plots of the gradient’s norm of the SARDF intersection, when the angle approach (second approach) is used in the third quadrant. Values of the radius (first quadrant) and the angle (third quadrant) are respectively: top left: 0.1, 0.0; top right: 0.1, 0.2; bottom: 0.1, 0.5
Figure 5.4, top left corresponds to radius of 0.1 and an angle of 0.0; Figure 5.4, top right corresponds to radius of 0.1 and an angle of 0.2. And finally, Fig. 5.4, bottom corresponds to radius of 0.1 and 0.5. Note that the maximum possible value for the angle has to be strictly less than $\Pi/4$. Table 5.2 contains some values of the gradient’s norm at different points and for different values of the parameters. The norm of the gradient is equal to 1 everywhere, but in the band where the circular approximations are performed. In this area, the value of the gradient’s norm is below 1. Modifications of the parameter in the third quadrant only (the angle) is considered as the influence of the radius in the first quadrant has been studied above. When increasing the opening angle for the circular approximation, the norm of the gradient decreases slightly from 1. Note that even for an angle of 0.5, the minimum value, reached on the diagonal $x = y$, is of 0.815471 approximately.

<table>
<thead>
<tr>
<th>parameters</th>
<th>0.1, 0.0</th>
<th>0.1, 0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>point</td>
<td>(-5.0,-5.0)</td>
<td>(-0.3,-0.3)</td>
</tr>
<tr>
<td>norm value</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>parameters</th>
<th>0.1, 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>point</td>
<td>(-1,-0.5)</td>
</tr>
<tr>
<td>norm value</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.2: Value of the norm of the SARDF gradient for different values of the parameters, when the second approach in the third quadrant for constructing the SARDF intersection is used.

These results allow us to make some recommendations on the choice of the approach and the parameters for the SARDF intersection. It seems better to use the second approach, with an opening angle in the third quadrant. The value of the angle depends on the object modeled and the primitives to which the SARDF intersection is applied. In the first quadrant it seems better to use a small maximum radius for the bounded circular approximation.

SARDF union and intersection present some symmetries ($\text{intersection}(-x,-y) = -\text{union}(x,y)$), so the same results would be obtained and the same recommendations
given, but with exchanging the quadrants 1 and 3. It seems better to use the second approach, with an opening angle in the first quadrant. In the third quadrant it seems better to use a small maximum radius for the bounded circular approximation.

**Conclusion:** The study of the norm of the gradient for the intersection function is, as underlined above only a particular case, but it provides already useful informations for drawing some conclusions.

Min has a unit norm gradient, when the gradient is defined; the gradient of the min function is not defined on the line \( y = x \); which when applied to functions corresponds generally to the set: \( \{ p : f_1(p) = f_2(p) \} \).

R-function has a unit norm gradient only on the L shape, the norm quickly decreases in the first quadrant and quickly increases everywhere else.

SARDF intersection has a unit gradient’s norm everywhere, but in the band where the circular approximation is computed. For some values of the parameters, in fact when the radius bound is big, the norm of the gradient can quickly drop (see in Table 5.1 a value of 0.182018 in the first quadrant for a radius of 5.0). However, a reasonable use of SARDF operations should use a low value for the bound of the growing radius. In the case of the intersection, the use of the angle approach in the third quadrant, with an angle 0.0, and a bound for the radius below 1.0 should give a good approximation. In the particular case of the L shape, the use of the SARDF intersection with an angle of 0.0 and a radius of 0.2 gives a very good approximation of the exact distance to the L shape; the distance field is exact in quadrant 3, and a good approximation in the quadrant 1 (and of course exact in quadrants 2 and 4).

### 5.1.3 Qualitative comparison of the union operations: SARDF union, max, and R-union

We compare the three union operations: max, R-union, and SARDF union applied in constructive modeling in terms of the quality of approximation of the Euclidean distance for the resulting function, and in terms of the smoothness of this function.
The solid used in the experiments is the union of two ellipsoids centered at \((5, 0, 0)\) and \((5, 0, -5)\) with respective radii \((5, 2, 2)\) and \((2, 2, 5)\).

Let \(f_1\) and \(f_2\) be the signed Euclidean distance functions defining the two ellipsoids. The interior of an ellipsoid corresponds to \(f_i > 0\) and the exterior to \(f_i < 0\). The surface boundary of an ellipsoid corresponds to \(\{p \in \mathbb{R}^3 : f_i(p) = 0\}\). The procedure used for computing the exact signed distance from a point in \(\mathbb{R}^3\) to the surface of an ellipsoid is a modification of the method proposed in [40].

We are interested in the quality of the approximation of the Euclidean distance function and the smoothness of \(g = f_1 \lor f_2\), where \(\lor\) stands for the functional definition of one of the three union operations. The expression for the SARDF union operation is constructed similarly to the SARDF intersection, as described in Section 3.1.3, and the expression used for the R-union of two objects is \(f_1 \lor f_2 = f_1 + f_2 + \sqrt{f_1^2 + f_2^2}\).

Figure 5.5 shows the contour maps of a cross-section by the plane \(y = 0\) of the resulting defining functions for the different analytical expressions for union.

Figure 5.5: Contour maps of the union of two ellipsoids for the three different union operations. The level of gray corresponds to ranges of distances from points inside the solid to its surface. Left: SARDF union, middle: max: sharp corners indicate a \(C^1\) discontinuity, right: R-union: contour map of the function clearly shows that the resulting function loses the distance like behaviour quite close to the boundary even with exact distance functions used for the arguments.
Figures 5.5 left, middle and right show interior contour lines of the function for the considered object. The figures correspond to the models made respectively with SARDF union, max, and R-Function union. The contour map of the defining function built with the R-union loses the Euclidean distance like behaviour quite close to the surface boundary even with exact distance functions defining the two ellipsoids. This loss of the distance like property is emphasized with the comparison to the maps given by using SARDF union and max (Fig. 5.5 left and middle). These contour maps have very similar contour lines except at the points joining the contour lines of the two functions, which are sharp for max (see Fig. 5.5 middle) and smooth for SARDF (see Fig. 5.5 left). It illustrates the discontinuity of the partial derivatives of the function defining the solid built using max.

Points of discontinuity of the Euclidean distance’s partial derivatives are also present on the main axis of each ellipsoid (Fig. 5.5). It is known that as soon as at least two points of the shape have equal distance values to the given point in space, the Euclidean distance function has a discontinuity of its partial derivatives at this point. This set of points is known as the medial axis of a shape. The discontinuity of the partial derivatives of the primitives at these points remains independently of the analytical expression used for the union operation.

5.1.4 Comparison of SARDF operations, min/max and R-functions with the exact distance function

Three shapes are considered: a cube (intersection of planes), an union of two spheres, and a “lozenge-like” (intersection of planes).

For each shape, there are two representations: a FRep and a mesh of triangles (BRep). The FReps are obtained by applying set-theoretic operations (SARDF, min/max or R-functions) on simple primitives defined by the Euclidean distance function (in these examples, only planes and spheres). The resulting FRep function defines for a model an approximation of the Euclidean distance field to the boundary shape.
The BRep (triangle mesh) for the union of spheres is created by polygonization [71, 12] of the FRep model. The BRep for the union of spheres is made of a lot of small triangles, whereas the two other shapes contain only a few number of triangles.

The signed Euclidean distance function to each BRep is computed on a 3D grid. The accuracy of the signed distance function depends of the tessellation accuracy (if the BRep is obtained by polygonization of a FRep) and the size of the grid’s cells. The Euclidean distance to a BRep is computed using the following algorithm:

**distance to mesh (point p, mesh m)**

1. Initialization. set \( \text{min} \) to infinity; search the nearest neighbors of \( p \) among the vertices of \( m \); set \( s \) to the set of triangles that have one of the nearest neighbors as a vertex;

2. Iteration. for every triangle \( t \) in \( s \) compute the distance between \( t \) and \( p \) and save: the minimum distance in \( \text{min} \), the corresponding triangle \( t \) in \( \text{closest\_triangle} \), and the corresponding closest point in \( \text{closest\_point} \);

3. Find the sign. set \( n \) to the normal of \( \text{closest\_triangle} \) at \( \text{closest\_point} \); set \( \text{scalar} \) to the scalar product between \( n \) and the vector between \( p \) and \( \text{closest\_point} \); if \( \text{scalar} > 0 \) then set \( \text{sign} \) to 1 otherwise to -1;

4. Return. return \( \text{sign} \times \text{min} \);

The point-set consisting of the vertices of \( m \) is stored in a kd-tree [7] to accelerate the search of the nearest neighbors of \( p \) in \( m \). Construction of a kd-tree has a complexity of \( O(N\log N) \) and queries have an average complexity of \( O(\log N) \), where \( N \) is the number of points in the mesh.

For the tests, a grid of size \( 100 \times 100 \times 100 \) is used; the bounding boxes for the cube, spheres, and lozenge are respectively: \([-1,-1,-1] \times [2,2,2] \), \([-2,-2,-2] \times [3,2,2] \) and \([-1.7,-2.1,-0.1] \times [2.5,2,2] \).
SARDF operations use the unbounded approach in the quadrant 3 (for the union, it is in quadrant 1) with an angle of 0.0, the radius for the bounded approach is 0.1. Min/max, R-Functions and SARDF models are evaluated on the same grid as the distance function and their values are compared.

To compare the different functions with the distance function, we compute the minimum, maximum, mean error and the variance of the error. The results are given in Table 5.3.

\[
\begin{array}{|c|c|c|c|c|}
\hline
& \text{min} & \text{max} & \text{mean} & \text{variance} \\
\hline
\text{SARDF cube} & 0 & 0.0614415 & 0.000201567 & 0.00205558 \\
\text{min cube} & 0 & 0.732051 & 0.108827 & 0.128018 \\
\text{R-function cube} & 5.21282e-16 & 13.4036 & 3.35523 & 2.33183 \\
\text{SARDF spheres} & 0 & 0.132702 & 0.00163311 & 0.00419018 \\
\text{max spheres} & 1.16907e-6 & 0.315158 & 0.00120212 & 0.00652017 \\
\text{R-function spheres} & 1.99219e-5 & 0.985591 & 0.289187 & 0.184477 \\
\text{SARDF lozenge} & 5.10165e-6 & 1.10966 & 0.400159 & 0.199127 \\
\text{min lozenge} & 1.57946e-7 & 1.05257 & 0.37966 & 0.173445 \\
\text{R-function lozenge} & 9.75467e-7 & 24.5256 & 2.59022 & 3.4196 \\
\hline
\end{array}
\]

Table 5.3: Min, max, mean and variance of the error between approximate distance functions created using SARDF operations, min/max, or R-functions and the distance function.

We consider also the following distance metric between a distance function approximation and the distance function, available on a grid:

\[
dist = \frac{1}{n} \sum_{i=1}^{n} (\text{approx}(i) - \text{exact}(i))^2
\]

(5.1.4)

where \(i\) is running over the voxel indices, \(n\) is the total number of voxels, \(\text{approx}(i)\) is one of the approximation built using min/max, R-functions or SARDF operations and normal primitives, and \(\text{exact}(i)\) is the signed distance function, as computed by the previous algorithm \textbf{distance to mesh}. The value of the distance metric for each object is given Table 5.4.
Table 5.4: The error from the distance function to each approximate distance using the error metric Eq. (5.1.4).

<table>
<thead>
<tr>
<th>Shape</th>
<th>distance metric (Eq. (5.1.4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>SARDF cube</td>
<td>4.26603e-6</td>
</tr>
<tr>
<td>min cube</td>
<td>0.0282318</td>
</tr>
<tr>
<td>R-function cube</td>
<td>16.695</td>
</tr>
<tr>
<td>SARDF spheres</td>
<td>2.02247e-5</td>
</tr>
<tr>
<td>max spheres</td>
<td>4.3957e-5</td>
</tr>
<tr>
<td>R-function spheres</td>
<td>0.117661</td>
</tr>
<tr>
<td>SARDF lozenge</td>
<td>0.199779</td>
</tr>
<tr>
<td>min lozenge</td>
<td>0.174229</td>
</tr>
<tr>
<td>R-function lozenge</td>
<td>18.4029</td>
</tr>
</tbody>
</table>

As written previously, functions built using R-functions explode numerically even for points close to the surface. It is less obvious in the case of the union of two spheres, because most of the sampled points are located within the two spheres, where the error due to the R-functions is geometrically bounded. Min/max and SARDF operations behave quite similarly with respect to the distance function. However, SARDF functions are smoother.

If a bounded radius approach were used in both quadrants for the SARDF operations, then the result would also match with min/max; a small value of the radius for the SARDF operations should be used. Using an angle approach, allows to better mimic the distance function in some cases; for example, in the case of the cube, we obtain the exact distance function, up to the corners smoothed within the cube. In our experiments, we used everywhere an angle $\alpha$ of 0.0. The value of the angle is related to the geometry of the surface of the solid.

The circular arc approximation, of the SARDF operations, is done in the "functional space", whereas a circular or spherical approximation should be done in the "geometric space" in order to obtain a better approximation of the distance function. It is however difficult to do practically.
So far, we have illustrated two properties: the SARDF operations do not introduce additional points where the resulting function is not $C^1$, like min / max; and they are better approximations of the Euclidean distance than the R-functions. Both these properties are important in solid modeling and applications as for example in heterogeneous object modeling: Shin and Dutta relate that R-function is not an exact distance function, consequently making it difficult for a designer to predict or control the material distribution (section 3.3, p. 210 and Figure 10, p. 211 of [97]). Biswas et al report that the lack of smoothness in a material function parameterized by the distance results in zones of concentration or stress of the materials (section 1.3 of [10]).

5.2 Constructive heterogeneous objects modeling with SARDF

We propose some examples to illustrate the use of the introduced SARDF operations and normal primitives in constructive heterogeneous modeling. The SARDF operations as described in Chapter 3, Section 3.1.3 are used instead of the R-functions or the min/max functions in the different constructive trees to define the geometry of the solid and the partitions where attributes are defined.

By normal primitive, we mean a primitive with a defining function $p$, which at a given point $\mathbf{x} \in \mathbb{R}^3$, returns the Euclidean distance, or an approximation, from $\mathbf{x}$ to the surface $p^{-1}(0)$. Primitives with known expressions for the distance are given in [39].

We show by plotting the functions defining the material distributions, how the different expressions for the set-theoretic operations affect the material distributions and their properties.

5.2.1 Two-dimensional example

At first, we illustrate the use of SARDF on a two-dimensional heterogeneous object. The geometry of the object (Fig. 5.6a) is defined as $f(\mathbf{X}) \geq 0$, where $f$ is evaluated
by traversing the constructive FRep tree [70] with a box and a cylinder in the leaves, and the subtraction operation in the node. Expressions for the distance function corresponding to the box and the cylinder can be found in [39]. The subtraction is defined with the SARDF intersection: 

\[ f_1 \triangleleft f_2 = f_1 \land (-f_2). \]

Figure 5.6: Two-dimensional CAD part. (a) Geometry of the CAD part defined by an FRep model. (b) Three different material regions (blue: material 1, red: material 2, color gradient: functionally graded material).

This object is made of two materials and three material regions (Fig. 5.6b). We use the notation \( m_1(X) \) and \( m_2(X) \) for the scalar volume fraction component of the materials 1 and 2. For visualization purposes, the material distributions are mapped to the "RGB" color space: a color is attributed to each material and the final color is the combination of the colors corresponding to each material, weighted by the scalar volume fraction.

Two of the three material regions (so called material features) correspond to spaces where: there is only material 1 uniformly distributed (blue in Fig. 5.6b) and there is only material 2 uniformly distributed (red in Fig. 5.6b). The last material region corresponds to the functionally graded material. The geometry of each region is defined using FRep in a constructive way, similarly to the shape’s geometry. SARDF operations are used in the nodes and normal primitives in the leaves of the constructive tree. The resulting functions provide \( C^1 \) approximation of the distance to each material region. These distances are used to specify the functionally graded material.

The scalar volume fraction of each component material in the functionally graded
material region is given by: \( m_1(X) = w_1(X)M_1 \) and \( m_2(X) = w_2(X)M_2 \), where \( M_1 \) and \( M_2 \) stand for the value of the scalar volume fraction on the boundary of the first and second material features shown respectively in blue and red Fig. 5.6b.

For the weighting functions \( w_1(X) \) and \( w_2(X) \), we use a normalization of each inverse distance functions:

\[
\begin{align*}
w_1(X) & = \frac{1}{\frac{1}{d_1(X)} + \frac{1}{d_2(X)}} = \frac{d_2(X)}{d_1(X) + d_2(X)} \\

w_2(X) & = \frac{1}{\frac{1}{d_1(X)} + \frac{1}{d_2(X)}} = \frac{d_1(X)}{d_1(X) + d_2(X)}
\end{align*}
\]

(5.2.1)

(5.2.2)

where \( d_1(X) \) and \( d_2(X) \) are the distances from point \( X \) to the boundary of respectively the material features shown in blue and red Fig. 5.6b.

These two distance maps are illustrated in Fig. 5.7 and Fig. 5.8. Fig. 5.7a and Fig. 5.7b correspond respectively to the approximate distance map \( d_1 \) when the R-functions and the SARDF operations are used correspondingly to define the shape. In a similar way, Fig. 5.8a and Fig. 5.8b correspond to the approximate distance \( d_2 \) when using R-functions and SARDF.

The approximate distance maps built using R-functions indicate that even if R-functions have good smoothness properties, they are not a good approximation to the distance function, making it difficult to control accurately material distribution.

The weighting functions \( w_1(X) \) and \( w_2(X) \) are continuous functions satisfying the interpolation condition \( w_i(\partial B_j) = \delta_{ij} \), where \( i, j \in 1, 2 \), \( \delta_{ij} \) is the Kronecker symbol\(^1\), and \( \partial B_j \) are the boundaries of the material features seen in blue and red Fig. 5.6b. The functions \( w_1(X) \) and \( w_2(X) \) form a partition of unity.

The properties of these functions are illustrated in Fig. 5.9 with a cross section of the model through the \( y-axis \) and the visualization of the evolution of the weighting functions \( w_1(X') = (x, \text{const}) \) and \( w_2(X') \) along the \( x-axis \). Note that in the current example \( w_i \) and \( m_i, \ i \in 1, 2 \), have the same graph, since the values of the volume fraction on the boundaries, \( M_1 \) and \( M_2 \) have been chosen equal to 1.

\(^1\)equals to 1 if \( i = j \) and 0 otherwise
Figure 5.7: Approximate distance map $d_1$ from point $X$ to the boundary of the region where only material 1 exists. (a) Using R-functions. (b) Using SARDF operations.

Figure 5.8: Approximate distance map $d_2$ from point $X$ to the boundary of the region where only material 2 exists. (a) Using R-functions. (b) Using SARDF operations.
Figure 5.9: A cross-section parallel to \( x \)-axis and the distribution of the materials in the cross section for the CAD part constructed with SARDF functions.
One can notice (Fig. 5.9) a $C^1$ discontinuity at the points on the boundary of the material features. This can cause the same problems as the distance function $C^1$ discontinuity. Fortunately, these sharp corners can be smoothened by a modification of the expressions for the coefficients (Eq. 5.2.1 and 5.2.2). Indeed the expressions used for the material feature weights correspond to a particular case of the inverse distance weighting [96]. More general expressions are: $w_1(X) = \frac{d^k_1(X)}{d^k_1(X)+d^k_2(X)}$ and $w_2(X) = \frac{d^k_2(X)}{d^k_1(X)+d^k_2(X)}$. The case $k = 1$ gives Eq. 5.2.1 and 5.2.2. The parameter $k$ controls the smoothness of the functions on the points of the material features.

Replacing every SARDF operation by an R-function or min/max in the constructive trees for the geometry of the solid and the material regions gives different material distributions in the same cross-section (see Fig. 5.10, right and 5.10, left).

Figure 5.10 reflects at the level of the material distribution the problems of using the R-functions (right graph), or min/max (left graph) in constructive heterogeneous modeling. Figure 5.10, right, shows the role played by the accuracy of the distance
approximation when the distance is used to parameterize the material distributions. The unpredictable behaviour of the distance approximation makes the task of the designer difficult. For example, we would expect that the first part of the blue curve (just before the intersection with the red curve) is linear. This "bad behaviour" of the R-functions was noticed by Shin and Dutta in [97].

Figure 5.10, left, illustrates the $C^1$ discontinuity of the min (and max) functions and its impact on the material distribution. Both distributions of material 1 (blue) and 2 (red) have two points of $C^1$ discontinuity (circled in Fig. 5.10, left). It results in problems of stress or concentrations as noticed by Biswas et al in [10].

Using SARDF for the set-theoretic operations does not introduce new points of $C^1$ discontinuity, and keeps a good approximation of the distance, these properties can be seen consequently in the graph of the material distributions (Fig. 5.9).

In this example (also in the following) only two materials are in the overlapping zone. More materials can be blended and the expressions for inverse distance weighting (Eq. 5.2.1 and 5.2.2) can be extended to the case where more than 2 materials are blended. Additional details on the inverse distance weighting used for the interpolation of materials defined over functionally defined sets can be found in [86]. More complex expressions for compositions of multiple materials, like vector valued materials, constrained and weighted interpolation of materials, can be found in [10].

The resulting model of the CAD part and its material distribution is illustrated by Fig. 5.11. The distribution of the material 1, given by its scalar volume fraction $m_1(X)$ is mapped to the blue color, and the distribution of the material 2, given by $m_2(X)$ to the red color. Stripes are used to make the visualization of the changes in material distribution easier.

### 5.2.2 Three-dimensional CAD part

We propose a second example (in three dimensions) with more complex shapes for the geometry of the object and the geometry of the regions corresponding to the material
Figure 5.11: Material distribution for the CAD part: the zone in blue contains only the material 1, the zone in red contains only the material 2, the zone in gradient corresponds to a region where both materials are blended, the weight of each material is given by Eq. 5.2.1 and 5.2.2.

The overall geometry of the object is a block with two (constant) material features inside. We keep the same notation as in the previous subsection, with \( m_1(X) \) and \( m_2(X) \) the scalar volume fraction of the materials 1 and 2. Figure 5.12a shows the first material feature corresponding to the material 1 (in blue); it is cut by a planar half-space for visualization purposes only. Figure 5.12b shows the second material feature (in red); its geometry is composed of blocks and ellipsoids, combined with SARDF unions and intersections. The right of Fig. 5.12b illustrates a zoom on one of the pins. Such a pin is modeled with ellipsoids as primitives and SARDF union and intersection as operations. Exactly, it is the SARDF union of four ellipsoids, which are after subtracted from a fifth ellipsoid.

To express the material behaviour in the region between the two material features (this region can be seen in Fig. 5.12c), we use the equations 5.2.1 and 5.2.2 for the weights for each material feature. It indicates that the closest material feature has the strongest influence. The overall distribution of the materials is shown in Fig. 5.13a. The geometry corresponding to the second material feature is rendered, using a red
Figure 5.12: (a) the first material feature, (b) the second material feature, with a zoom on one of the pins, on the right, (c) union of the two material features (Model made by Benjamin Schmitt).
color, then for the visualization of the material distribution, two cross-sections are made: one for $x = 0$ and one for $z = 0$. For each of the cross-section, the evolution of the material distribution is projected. For visualization purposes, each material is mapped to one color. The first material corresponds to the blue color and the second material to the red.

5.3 Constructive modeling with SARDF: applications

Distance functions and their approximations have many applications in shape modeling. We illustrate here metamorphosis, constant-radius offsetting, and tapered extrusion as applications of SARDF in animation and design.

The SARDF operations used follow the construction given in Chapter 3, Section 3.1.3.

5.3.1 Metamorphosis of functionally defined shapes

We use the metamorphosis example from [38] to illustrate the use of SARDF objects in metamorphosis of functionally defined shapes. The paper proposes a metamorphosis between a Buddha shape and a kanji shape. Both objects are functionally defined, and the metamorphosis operation is a linear interpolation between the initial and final shape. Suppose $f_i$ is the defining function for the Buddha and $f_f$ the defining function for the kanji, then the metamorphosis from $f_i$ to $f_f$ is given functionally by: $\text{metamorphosis}(x, y, t) = (1 - t)f_i(x, y) + tf_f(x, y)$, with $t \in [0, 1]$ representing the time. Figure 5.14 shows the initial (Buddha) and final (kanji) shapes of the metamorphosis.

Both shapes are given by polylines with straight segments. The FReps for both the Buddha and the kanji shapes are constructed by applying set-theoretic operations (union and intersection) to the halfplanes bounded by straight lines going through the polyline segments. Rvachev et al [85] proposed an algorithm for converting a set of polygons to a constructive tree consisting of halfplanes in the leaves and set
Figure 5.13: Distribution of two materials. Blue color corresponds to material 1, red color to material 2. The color variation indicates the fraction of each material. (a) Two cross sections are made for $x = 0$ and $y = 0$ to show the material distribution. (b) A zoom is made on one of the pins with two additional cross-sections.
operations in the internal nodes, this algorithm was later extended by Shapiro to incorporate circular arcs [93].

In the original paper describing the metamorphosis from the kanji to the Buddha shape, R-functions are used for defining the set-theoretic operations in the internal nodes of the tree; it results in huge values of the final function defining the shape. A sampling of the function on a 200 by 200 grid reveals that the function defining the Buddha can reach $10^7$ outside. This explosion of the function value due to the use of R-functions poses some problem for the animation.

![Figure 5.14: Left, a Buddha shape, initial shape for the metamorphosis; right, the final shape for the metamorphosis: a kanji, meaning Buddha](image)

The high slopes in function values are responsible for unaesthetic and difficult to control intermediate shapes during the animation. See Fig. 5.15 for the intermediate frames of the animation built using R-functions, and Fig. 5.16 for the intermediate frames of the animation built using SARDF operations. In Frames 2 and 3, Fig. 5.15, the left leg of the Buddha shape starts to morph, but the right stays unchanged because of the unbalanced function values in these two areas; see by comparison the frames 2 and 3 of Fig. 5.16. The use of SARDF operations allows for predictable and controllable defining functions.

At some point, one may want to scale one or both of the functions $f_i$ or $f_t$ to modify the shapes of the intermediate frames of the metamorphosis. When R-functions are used, the high jumps in function values make it more difficult to predict the behaviour
Figure 5.15: Intermediate frames of the metamorphosis between the Buddha shape, and the kanji with R-functions used as set-theoretic operations during modeling.

Figure 5.16: Intermediate frames of the metamorphosis between the Buddha shape, and the kanji with SARDF functions used to implement set-theoretic operations during modeling.
of the function and to select a proper scaling function.

5.3.2 Constant radius offsetting

Constant radius offsetting was firstly proposed in [72] for functionally defined shapes. Their algorithm follows [81]. Given a function $f$ defining a solid by $f \geq 0$, and the radius for offsetting $R$, the algorithm for constant radius offsetting in the positive direction consists of the following 3 steps:

1. sphere definition: for any point of evaluation $p$, define a sphere centered in $p$ and of radius $R$

2. maximization: find $F$ maximum value of $f$ on the sphere

3. evaluation: return $F$ as a value for $f(p)$

When a constant radius offsetting in the negative direction is needed, the second step is replaced by a minimization on the sphere.

The defining function $f$ is usually non-linear, and the search of the global extremum on the sphere requires numerical methods. We use the conjugate gradient method, with the starting point obtained by sampling the function on the sphere. In our experiments, the sphere was initially sampled 380 times ($20 \times 19$). It seems possible that this figure can be decreased but we have not experimented with it yet. We experimentally found that in two dimensions, the sampling produces good enough results without using other numerical methods. In three dimensions, however it is not sufficient and a numerical algorithm is needed. Using only sampling in three dimensions produces poor and noisy results after polygonization.

Non-linear optimization requires usually the partial derivatives of the function to be optimized. We compute them analytically, because we found that numerically computed partial derivatives produce poor results. Neither R-functions nor SARDF functions are differentiable in $(0, 0)$, we compute however a gradient by averaging the gradients of the functions, arguments of the R-functions or SARDF, and normalize it. The search is performed on the sphere by penalizing points away from the sphere.
We present the results and time complexity for two models: a cube and a more complex L shape (made of convex and concave polygons) see Fig. 5.17. Both objects were offsetted in the positive direction. The cube has an offset radius of 5, and the L shape an offset radius of 11. The code has not been optimized. The timing on a SGI Fuel gives the following results for a polygonization done by Dual Contouring [48] with a grid of $50^3$: cube (R-functions) 296.3 seconds, cube (SARDF) 189 seconds, L shape (R-functions) 384.22 seconds, L shape (SARDF) 300 seconds. It is interesting to notice that constant radius offsetting is faster when SARDF is used in modeling, even if SARDF operations are more expensive to compute than R-functions. Gradient methods have sometimes problem to select the step size along a selected direction because there is no connection between the partial derivatives values and the distance in space; the use of a better approximation of the distance, provided by the SARDF, helps there.

![Figure 5.17: Constant radius offsetting for an L shape consisting in concave and convex polygons](image)

### 5.3.3 Tapered extrusion

The extrusion operation is usually given by the generator (planar region on the xy-plane) and the distance of translation in the orthogonal direction (parallel to z-axis). All points swept by the generator during this motion make an extruded object. However, in CAD applications for technological purposes one more parameter is added,
which is called the taper angle. The generator during the sweeping changes its shape (see Fig. 5.18). Let us consider an arbitrary cross-section by a plane orthogonal to both xy-plane and the generator boundary (see cross-section A) at some point. In the xz-view (right in Fig. 5.18), it has not a vertical edge parallel to z-axis (as in standard extrusion), but a tilted edge with angle $\alpha$ to the vertical z-axis. For example, in plastic molds the taper angle of five degrees is typically applied. Note that this constant angle effect cannot be achieved for the entire model with the traditional tapering operation, where the scaling factor is linearly changed along the z-axis.

Suppose the linear size of the initial generator (in some cross-section) is $L_1$, and the height of the extrusion is $H$, then the size of the generator at the height $H$ is $L_2 = L_1 - 2d$, where $d = H \tan(\alpha)$. Therefore, given the angle $\alpha$, we can calculate the total distance offset for the generator in the given tapered extrusion. The result of such a tapered extrusion operation can be described as follows:

$$F(x, y, z) = F_{\text{cgen}}(x, y, z) \land_S z \land_S (H - z),$$

(5.3.1)

where $F_{\text{cgen}}(x, y, z) = \text{Offset}(F_{\text{gen}}(x, y), d(z))$ is a negative offset by $d(z)$ along the normal to the initial 2D generator defined by the function $F_{\text{gen}}(x, y)$, the offset distance value $d(z)$ is a linear function:

$$d(z) = z \tan(\alpha),$$

(5.3.2)
and $\wedge_S$ stands for the SARDF intersection operation. If $F_{gen}(x, y)$ is a signed distance function, then offset along the normal is equivalent to the iso-valued offset and

$$F_{cgen}(x, y, z) = F_{gen}(x, y) - z \tan(\alpha)$$  \hspace{1cm} (5.3.3)

Let us apply SARDF operations to define the constructive generator such that the function $F_{gen}(x, y)$ closely approximates the distance function and the above tapered extrusion can be implemented.

Figure 5.19: Tapered extrusion of a convex generator (top, left) modeled with R-functions (top, right), min/max (bottom, left), and SARDF operations (bottom, right)

The first example shown in Fig. 5.19 presents the tapered extrusion by the convex generator with the polygonal boundary. Three shown 3D solids are swept by the
generator modeled with R-functions, min/max, and the proposed SARDF operations respectively (top right, bottom left, and bottom right in Fig. 5.19). It is clear that the R-functions do not provide the expected result as they do not approximate the distance function. The model with min/max operations is correct, but the defining function is not differentiable at many points, which creates problems for further applications of this function as it was described before. The model with SARDF operations looks similar and the function is differentiable everywhere. The edges of the model are blended with the given radius of the circular arc in the SARDF operation (in Fig. 5.19 this radius is made big enough in respect to the model size for illustration).

Figure 5.20: Tapered extrusion of the concave generator with the hole: generator (top left), contour map of its defining function (top right), and tapered extrusion (bottom)

The next example represents the tapered extrusion of the concave generator with the hole modeled with SARDF operations (Fig. 5.20). In this case, the radius of the circular arc in the SARDF operations is relatively small and the edges look sharp, however the defining function is differentiable and well approximates the distance function (see its contour map in Fig. 5.20).
5.4 Experiments with fitting template FRep models

Fitting template models is illustrated in the following through several examples. At first template models for a mechanical part and a sake pot are fitted to point-sets. The properties of fitting SARDF models to a point-set are then illustrated. Finally, application of template models fitting for finite element remeshing is illustrated.

5.4.1 First CAD part

The first test part contains 10714 points scattered on the surface. The input point-set is given in Fig. 5.21.

![Figure 5.21: Data set used for the reverse engineering of the mechanical part. It contains 10714 3D points, scattered on the surface of the part.](image)

The FRep defining function $F$ shown below is used as a parameterized model for the recovery process:

$$
\begin{align*}
 f(x, a) &:= (box(x, a) \setminus cylinderZ(x, a)) \\
 &\setminus cylinderZ(x, a);
\end{align*}
$$

(5.4.1)

This FRep model consists of three simple primitives: one box and two infinite cylinders oriented along the Z axis, each primitive is defined by its parameterized model. For example, in the case of the cylinder, the defining function is: $cylinderZ(x, a) := a[1] - \sqrt{((x[1] - a[2])^2 - (x[2] - a[3])^2)}$, where $a[1]$, $a[2]$, and $a[3]$ are parameters.
meaning the radius and a point on the $x - y$ plane, through which the axis of the cylinder passes. All the primitives are combined together using the R-Subtraction operator $\setminus$, which is itself defined analytically as discussed in [82] and [70].

\[
\begin{array}{|c|c|}
\hline
\text{set1} & \{-5, -4, -2, 10, 5, 4, 5, 3, 1, 3, -2, 1\} \\
\hline
\text{set2} & \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1\} \\
\hline
\end{array}
\]

Table 5.5: Two sets of initial values used for fitting: set1 is close to the best set of values, in contrary to set2.

Twelve parameters are released in the model, which is the maximum number in that case; they correspond to the lower-left corner of the box, three dimensions of the box, and the center and radius for each of the cylinders. The fitting algorithms need an initial estimation for the parameters, in the tests we use two sets of initial values configurations, they are given in the Table 5.5: one is close to the best fit (set1), in contrary to the second one (set2).

\[
\begin{array}{|c|c|c|c|c|}
\hline
& \text{Time in sec} & \text{Least square error} \\
& \text{set1} & \text{set2} & \text{set1} & \text{set2} \\
\hline
QN & 1.852 & 9.643 & 5.47 & 595.04 \\
SA & 1635.09 & 1773.109 & 5.49 & 5.49 \\
SAQN & 72.042 & 144.177 & 5.47 & 5.47 \\
\hline
\end{array}
\]

Table 5.6: Time (in seconds) taken by each method to converge to the best fit and least-square error (sum of the deviations squared) of the best fit for each set of initial values. QN stands for Quasi-Newton, SA for simulated annealing and SAQN is an hybrid algorithm consisting in a combination of simulated annealing and Quasi-Newton.

The results of the tests are given in terms of the following: least square error of the reconstructed model for the three methods: Quasi-Newton (QN), simulated annealing (SA), and hybrid method simulated annealing – Quasi-Newton (SAQN).
(see Table 5.6), time given in seconds taken to converge to the best fit for each of these methods (see Table 5.6), and the visual shape of the best fit (Fig. 5.22).

Figure 5.22: Shapes for the best fitted FRep in two cases: (right) the best fitted object does not correspond to the real object when starting with set2 and using the QN method; (left) the best fitted object corresponds to the real object when starting with the set2 and using the hybrid method.

Table 5.6 shows that the local method stops in a local minimum for the set2 of initial parameters, resulting in a wrong shape (Fig. 5.22, right shape), whereas with simulated annealing, it always converges to the global minimum. Unfortunately, the counterpart is the slow rate of convergence for the sampling method (Table 5.6).

When using a combination of SA and QN, we can recover correct parameters and shape (Fig. 5.22, left shape, and Table 5.6, last line). The steps of the shape evolution during the hybrid method search are shown in Fig. 5.23. Experimentally, a combination of SA and QN avoids local optima with a better convergence rate than SA alone.

5.4.2 Lacquer ware sake pot

The next example is the fitting of a model of a hand-crafted lacquer ware pot, which is used for pouring sake (Japanese rice wine). The discrete data set of the sake pot includes 27048 3D points, scattered on the surface of the object. The parameterized model of the sake pot sketched and discussed in the work on cultural heritage [73, 108] is reused in our experiment. The parameterized model was created using hand measurements of a typical sake pot. The major parameters are the coordinates of the
1. The origin (position), the basic radius of the pot body, and the height of the pot handle. The model is required to keep basic ratios of the measured sample object and to proportionally change the dependent parameters like those of the blend area between the spout and the body, and the shape of the lid holder (note non-linear changes of these shapes in Fig. 5.25).

2. Table 5.7: Initial and final vectors of parameters obtained by fitting the sake pot model with a Quasi-Newton method.

<table>
<thead>
<tr>
<th>initial vector of parameters</th>
<th>0</th>
<th>0</th>
<th>5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>final vector of parameters</td>
<td>2.705</td>
<td>0</td>
<td>4.804</td>
<td>7.71</td>
</tr>
</tbody>
</table>

The first fitting test is made using a Quasi-Newton algorithm. The results for the final vector of parameters are given in the Table 5.7. The value of the fitness function is big enough to indicate that the method stopped at a local minimum. A comparison of the discrete model and the fitted parameterized FRep, illustrated by Fig. 5.24, indicates that the best fitted parameters correspond to a local minimum.
of the fitness function.

![Image of a sake pot](image)

Figure 5.24: Local minimum effect: result of the fitting with a local method starting with wrong initial parameters. The point-set for the sake pot is also displayed for comparison.

The hybrid algorithm described before is used to go to the vicinity of the global optimum. The initial vector of parameters at the start of the process is the same as the one indicated in Table 5.7 (first line). In our experiments, the algorithm is stopped after the value of the fitness function goes below a threshold given by the user. In our tests, a value of 1000 for the fitness function is used as a threshold value to determine the final switch to a Quasi-Newton method. This value corresponds to an average error of 0.04 of the fitness function, and the current shape is confirmed by visual feedback. Then, the obtained parameters are reused as initial values for a final call to the Quasi-Newton method. The steps of the evolution of the shape during the hybrid fitting of the FRep model can be seen at Fig. 5.25.

The final result for the best fitted parameters obtained after the two steps of the hybrid method is summarized in Table 5.8 with total time for the fitting, the fit function value and the mean error.
Table 5.8: Final vector of parameters after using the hybrid approach: a global minimization method (simulated annealing) followed by a local minimization method (Newton).

<table>
<thead>
<tr>
<th></th>
<th>2.999</th>
<th>2.999</th>
<th>5.5</th>
<th>9.7044</th>
</tr>
</thead>
<tbody>
<tr>
<td>final vector of parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fitting time</td>
<td></td>
<td></td>
<td>347 sec</td>
<td></td>
</tr>
<tr>
<td>fit function</td>
<td></td>
<td></td>
<td>438.52</td>
<td></td>
</tr>
<tr>
<td>mean error</td>
<td></td>
<td></td>
<td>0.03</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.25: Evolution of the shapes of the sake pot during the fitting process
5.4.3 Fitting template models made with SARDF

The examples described above use R-functions for the set-operations. They also don’t systematically use Euclidean distance based primitives. As it was previously noticed in [24], it is better to fit models defined by the Euclidean distance. Following this remark, let us compare the results of fitting a template parameterized model defined using: R-functions and algebraic primitives, R-functions and distance primitives, min/max and distance primitives, and SARDF objects – i.e. SARDF operations and distance primitives.

The reconstructed object is illustrated by Fig. 5.26. The point-set used in the fitting has 10000 points scattered on the surface of the object. The selected parameterized model is made of planes, a cylinder and a sphere. There are 11 parameters in the model, corresponding to the positions and radius of the cylinder and the sphere, and to the position and length of the box.

![Figure 5.26: The reconstructed object](image)

Tables 5.9 and 5.10 give the results of fitting the different models made using:
| parameter 1     | 100003E+02 | 100035E+02 |
| parameter 2     | -152712E-03 | -377946E-02 |
| parameter 3     | 100002E+02 | 100030E+02 |
| parameter 4     | 299946E+01 | 300461E+01 |
| parameter 5     | 500031E+01 | 500031E+01 |
| parameter 6     | 499998E+01 | 499998E+01 |
| parameter 7     | 998585E+00 | 998631E+00 |
| parameter 8     | 148935E-03 | 13215E-03 |
| parameter 9     | 217971E-03 | 136767E-03 |
| parameter 10    | 266344E-03 | 232407E-03 |
| parameter 11    | 999948E+01 | 999957E+01 |
| least square error | 0.7727377854041829E-01 | 0.6207711536916782E-01 |

Table 5.9: Best fitted parameters and squared error for a template model made by: (1) R-functions and algebraic primitives and (2) R-functions and distance primitives

| parameter 1     | 100068E+02 | 100018E+02 |
| parameter 2     | -730635E-02 | -180436E-02 |
| parameter 3     | 100046E+02 | 100017E+02 |
| parameter 4     | 300922E+01 | 300190E+01 |
| parameter 5     | 500029E+01 | 500031E+01 |
| parameter 6     | 499998E+01 | 499998E+01 |
| parameter 7     | 998714E+00 | 998599E+00 |
| parameter 8     | 197011E-03 | 552494E-04 |
| parameter 9     | 806218E-03 | 578662E-04 |
| parameter 10    | -267563E-04 | 947201E-04 |
| parameter 11    | 999845E+01 | 99982E+01 |
| least square error | 0.17615955305276379E0 | 0.16018136029988042E-01 |

Table 5.10: Best fitted parameters and squared error for a template model made by: (3) Min/max and distance primitives and (4) SARDF operations and distance primitives
(1) R-functions and algebraic primitives, (2) R-functions and distance primitives, (3) min/max and distance primitives and (4) SARDF operations and distance primitives.

The given results consist in the values of the best fitted parameters, and the least square error – i.e. the sum of the squared deviations. These values of the parameters have to be compared with the values of the real object, which are: 10, 0, 10, 3, 5, 5, 1, 0, 0, 0, 10.

The reconstructed object obtained using the SARDF model and the best fitted parameters of the corresponding column from table 5.10 is given in Fig. 5.26.

The conclusion is that the SARDF model (4) gives the best results. Surprisingly, the results obtained when using min/max and distance primitives are the worst (3); they are even worse than the results obtained by using R-functions with algebraic primitives (1).

5.4.4 Application in Finite Element Meshes (FEM)

Approaches to FEM generation  Surface remeshing is very important for applications associated with numerical simulation procedures, in particular with finite element analysis (FEA). These applications impose strict constraints on the quality of the surface approximation and the shapes and sizes of mesh elements. Moreover finite element meshes have to be adapted both to physical and geometric features of computational tasks. Changes in the boundary or initial conditions of the simulated process may cause remeshing even if the computational domain remains the same.

In many cases the initial description of computational domains in FEA is represented by their boundary surface triangulations. These triangulations can be exported from various modeling systems, produced by 3D scanning, or be a result of previous FE computations. Usually these initial triangulations consist of badly shaped triangles and are not adapted to physical conditions and an appropriate remeshing is required. Mesh refinement and optimization procedures need accurate information about the geometry of the computational domain. Therefore, the creation of an adequate description of a solid based on the initial triangulation of its boundary surface
is an important problem for the FE mesh generation and optimization. Different approaches were considered to solve this problem. In [33], finite element adaptation is based on the local approximation of the underlying surface geometry by a quadric surface. The authors of [51] convert a CAD model into a volume representation by sampling its distance field on a uniform grid and then applying the extended marching cubes algorithm to this volume.

Taking into account that many mechanical parts can be represented as constructive solids we propose to apply FRep recovery to support FE mesh generation for objects whose initial geometry is represented by boundary surface triangulations. The initial mesh is used for the selection or creation of a parameterized FRep model. Then, the parameters of the FRep model are fitted to the vertices of the mesh. The final model can be used for the surface and volume finite element adaptation by the methods described in [49].

**Fitting to a CAD mesh** As an example of application of the FRep shape recovery for the FEM generation, a parameterized FRep model corresponding to the CAD mesh Fig. 5.27 (top, left) is created and fitted using the previously proposed techniques.

<table>
<thead>
<tr>
<th>fit function</th>
<th>0.667</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean error</td>
<td>0.011622</td>
</tr>
</tbody>
</table>

Table 5.11: Fitting function value for the best fit set of parameters and the corresponding mean error.

The FRep model including 14 parameters is sketched corresponding to the shape shown in Fig. 5.27, top, left; the initial values for the parameters are chosen randomly.

The convergence is obtained using an hybrid simulated annealing/Quasi-Newton scheme. The results in terms of the fitness function value and the mean error are presented in Table 5.11. The FRep shape corresponding to the best set of parameters is shown in Fig. 5.27, top, right.

Starting with the acquired FRep model, it is then possible to apply the mesh
adaptation methods from [49]. The results of such methods are shown in Fig. 5.27, bottom. The left picture shows an optimized surface mesh, which was then used for the 3D tetrahedral mesh generation (right) using the extended advancing front method [49].

Figure 5.27: A surface mesh, generated by a CAD system (top, left), the recovered shape (top, right), the associated optimized mesh (bottom, left), and the 3D tetrahedral mesh generated from it (bottom, right). The original BRep model and the generated optimized and 3D tetrahedral meshes are courtesy of Elena Kartasheva.
5.5 Constructive tree recovery using a genetic algorithm

The genetic algorithm, described in Chapter 3, Section 3.3, with details of implementation given in Chapter 4, Section 4.5, is applied to recover a constructive tree of some mechanical parts.

The input of the algorithm is a set of points scattered on the surface of the solid and the primitives associated to a segmentation of the point-set. Point-set segmentation and fitting of primitives to the different subsets of the segmented point-set are obtained by a brute force algorithm. A genetic algorithm is run on a list of primitives (cube, box, sphere, torus, ellipsoid), the best fitted primitive is selected and the corresponding points from the point-set are removed, then we loop to the first step until the point-set contains only a few points.

Primitives are implemented by the Euclidean distance function and operations by the SARDF functions.

5.5.1 Example 1

The first example consists of a point-set of 9530 points and 10 primitives. These primitives include planes, spheres and cylinders and were recovered using a brute force genetic algorithm as described above. In fact, a box was fitted instead of planes, but we decided to use the planes instead to increase the number of primitives. It should be noticed that the use of a different segmentation algorithm may have produced such planes instead of a box.

We used the genetic algorithm and the genetic operations described in Chapter 3, Section 3.3 and Chapter 4, Section 4.5. We used a probability of 0.1 for the mutation and of 0.6 for the crossover. Figure 5.28 illustrates the convergence after 200 generations of an initial population of size 1000.

The solid resulting from the best individual is given in Fig. 5.29.
Figure 5.28: Evolution of the fitness of the best individual for a population of size 1000

Figure 5.29: Result for the best individual for the first mechanical part
5.5.2 Example 2

The second example consists of a point-set of 49388 points segmented in 10 primitives. The same parameters as above are used for the size of population, probability of crossover and probability of mutation. Figure 5.30 illustrates the convergence after 200 generations of a population of size 1000. The solid obtained from the best individual is illustrated by Fig. 5.31.

![Evolution of fitness with number of iterations](image)

Figure 5.30: Evolution of the fitness of the best individual for a population of size 1000

![Result for the best individual for the second mechanical part](image)

Figure 5.31: Result for the best individual for the second mechanical part
5.5.3 Comments

Comparing figures 5.28 and 5.30, we find a higher fitness value for the second example. It can be explained by a bigger number of points in the point-set used: the second point-set is approximately 5 times bigger. It can also be explained by a different approximation in fitting the primitives.

The number of points in the point-set has some importance. If the number is too big, it will slow down the evaluation of the fitness function and the speed of the algorithm.
Chapter 6

Conclusion

6.1 Main contributions

In this document, a framework for the construction of volumetric objects using distance-based scalar fields has been presented. This framework is named SARDF, for Signed Approximate Real Distance Function and is a subset of the Function Representation (FRep) model [70]. Within this SARDF framework, an object is modeled in a constructive way by applying operations (SARDF operations) on primitives (defined by the Euclidean distance function or its approximation). Primitives and operations are defined functionally, with a closure property: the final object, obtained by applying operations to functions, is also described by a function. We review and detail in the following the main contributions of this thesis.

**SARDF operations** Extra requirements on the overall defining function are imposed: its value should return an approximation of the signed Euclidean distance function, and it should be smooth. The traditional implementations of set-theoretic operations, namely the R-functions and min/max, either suffer from inaccurate Euclidean distance approximation (R-functions), or are not smooth and create discontinuities of the partial derivatives of the constructed function (min/max). Therefore, we have introduced new functions, and presented their constructions and implementations for defining the set-theoretic operations – intersection, union and difference
– named SARDF operations that keep a reasonable approximation of the distance function and are smooth. We have proved that SARDF operations are $C^1$ functions.

**Primitives definition** For the overall defining function to behave like the distance function, the primitives need to have this distance property too. The construction of the Euclidean signed distance function has been given and detailed for various primitives, ranging from the natural quadrics to more complex freeform primitives.

**SARDF Framework** An implementation of the complete framework within the HyperFun project has been presented. The HyperFun interpreter has been modified and extended to allow modeling with SARDF. A new version of the HyperFun applet has been presented which allows the selection between Euclidean distance-based modeling and modeling using arbitrary real functions. In the former, an implementation in Java of the SARDF operations is used with other allowed operations and a set of primitives. Whereas in the latter, the applet is run in normal mode – it means R-functions or min/max are used with the normal HyperFun library of functions and operations.

**Constructive heterogeneous object modeling with SARDF** We have proposed to use the SARDF framework for constructive heterogeneous object modeling as an extension of the constructive hypervolume model from [69]. The distance function is used here to parameterize the material distribution [10]. We have illustrated through case studies the properties and importance of distance and smoothness of SARDF in constructive heterogeneous object modeling.

**Applications of the SARDF framework in shape modeling** We have proposed and presented some applications of the SARDF framework in this document to illustrate its properties; typically these are applications, which benefit from the distance property and/or smoothness of the function defining the solid. The first application is a linear metamorphosis between SARDF objects; we have illustrated
through an example of metamorphosis between a Buddha and a kanji, that the distance approximation for the solids’ definition allows a better control of the metamorphosis. The second application is the constant radius offsetting of SARDF objects. Two examples have illustrated that the use of SARDF objects speeds up the offsetting operation. The third application, tapered extrusion, requires a good distance approximation and smoothness of the defining function to give good results.

The SARDF framework is a constructive framework. The creation of constructive models is a difficult and tedious task. In order to automate this process, we have proposed to use template FRep models and presented algorithms for fitting them to point-set.

**Template models and fitting** Some objects belong to more general families of similar objects: for example several vases may possess the same structure and overall shape. Once a model has been made for an object, it can be abstracted by a template model, where each of the most important parameters have no specific values and will be fitted to adapt the model to other instances of objects. Taking again the example of a vase, a different vase can have a bigger body for example. We have introduced new algorithms for fitting template FRep models, which rely on the combination of heuristic methods – such as simulated annealing or genetic algorithms – with gradient based methods – such as Quasi-Newton or Levenberg-Marquardt.

Template models and the proposed fitting algorithms have been illustrated by fitting different examples of template models for mechanical parts and a more complex sake pot to discrete point-sets. A different mechanical part was used to illustrate an application of fitting template models in remeshing for finite element applications. The advantage of using SARDF operations with distance-based primitives, instead of R-functions or min/max, for defining the template model has been illustrated by fitting a mechanical part.
Constructive models recovery Finally, we have introduced a new algorithm for the recovery of constructive FRep models from a segmented point-cloud and the associated set of primitives. The search is performed by a genetic algorithm which looks for the best expression involving the primitives. The proposed algorithm is not restricted to the set-operations, and can be extended, by using blending operations.

From the recovered constructive FRep, it is possible to make a parameterized FRep model that can be reused in fitting, mesh optimization or other operations.

6.2 Future works and possible extensions

The modeling process can be greatly automated by reuse of template models and their optimization. The idea of template model can be extended in some directions: in this document, we have focused only on the optimization of the shape minimizing the distance to a point-set; but different optimizations and objective functions can be experimented leading to more general shape optimization.

The idea of template parameterized FRep leads to the more general idea of parameterization of the attributes and the automation of the properties (attributes) modeling. Similarly to the parameterization of the general shape, the geometry associated to the attributes can be parameterized and fitted to satisfy some modeling constraints. Using SARDF modeling to define the models allows to use the distance to the models’ shape and features as a natural parameter for the optimization.
Acknowledgments

First of all, I want to thank my research advisor: Professor Nikolay N. Mirenkov for his good advices throughout my research. I am grateful to Professor Alexander Pasko, who discussed with me several ideas and gave me good advices. I want also to thank various colleagues, who worked with me or gave me some advices during my research: Yuichiro Goto, Professor Christophe Rosenberger, Benjamin Schmitt, Professor Lothar Schmitt, Professor Christian Toinard, and Carl Vilbrandt.

I want to thank the members of the laboratory (former and present), especially Professor Rentaro Yoshioka and Yutaka Watanobe, for their help.

I am very grateful to the Japanese ministry of Education, Culture, Sports, Science and Technology for a scholarship.

I want to thank the friends, who supported me here in Japan: the Aotsu family, the Takeda family, the Kobayashi family, Philippe and his wife Keiko, Hiroko-san, Sakae-san as well as Mr Izumi. I also need to thank my friends in France: Cyprien, Francois, Philippe and Vincent.

My biggest thanks are of course for my wife Asuka, and my family in France and in Japan.


[61] Z. Michalewicz, Genetic algorithms + data structures = evolution programs, Springer Verlag, 1996.


