Skolkovo Institute of Science and Technology

TOPOLOGY AND PARAMETER OPTIMIZATION FOR ADDITIVE MANUFACTURING BASED ON FUNCTION REPRESENTATION

Doctoral Thesis

by

DMITRY POPOV

DOCTORAL PROGRAM IN ENGINEERING SYSTEMS

Supervisor
CREI Director, Professor, Iskander Akhatov

Co-Supervisor
Senior Research Scientist, Ph.D., Alexander Pasko

Moscow – 2022

I hereby declare that the work presented in this thesis was carried out by myself at Skolkovo Institute of Science and Technology, Moscow, except where due acknowledgement is made, and has not been submitted for any other degree.

Candidate (Dmitry Popov)
Supervisor (Prof. Iskander Akhatov)

Abstract

Additive manufacturing opened a new vision of design. This manufacturing technology provides extra freedom in possible shapes and structures. However, new opportunities arose new challenges for design tools. Problems with precision, correctness, interoperability, and standards became highly sensitive. Moreover, this freedom forces people to choose from more design options. It leads to a new need that is addressed in modeling systems. Users want a software to help them with their choice.

The core of this work consists of a computer-aided design and manufacturing system development suitable for the needs of additive manufacturing. It analyses existing solutions and problems and proposes an architecture of the system and its prototype. The thesis describes the modeling and manufacturing framework for 3D printing technology. Its routines exploit a unique representation of the geometry, function representation. It allows avoiding problems with converting models in different formats. Therefore, all modeling, optimization, and control program generation for 3D printing works with one geometry representation.

The modeling component of the system is implemented as a web-based service. It provides users opportunities for collaborative work. The geometric core based on function representation allows storing models in a compact symbolic form. The thesis proposes rendering algorithms for these models, which rely on parallel computation with a graphics processing unit performed in a browser.

The optimization component of the framework uses an updated topology optimization algorithm based on the level-set method. Its modifications allow performing an optimization faster. Moreover, the thesis discusses aspects of including shape constraints in the optimization process, parameter optimization of a model, and their combination in one optimization process.

The manufacturing component of the proposed system exploits time-efficient algorithms based on exhaustive enumeration, interval arithmetic, and affine arithmetic to prepare parts for 3D printing. They are flexible for different scales and structural complexity of the models. The described component allows users to generate control programs for 3D printing equipment with open protocols.

The developed system was applied for the design and manufacturing of several example parts. Mechanical tests of these parts were performed to validate the developed system.

Publications

Main author

1. Dmitry Popov, Evgenii Maltsev, Oleg Fryazinov, Alexander Pasko, and Iskander Akhatov. Efficient contouring of functionally represented objects for additive manufacturing. *Computer-Aided Design*, 129:102917, 2020b

2. Dmitry Popov, Yulia Kuzminova, Evgenii Maltsev, Stanislav Evlashin, Alexander Safonov, Iskander Akhatov, and Alexander Pasko. CAD/CAM system for additive manufacturing with a robust and efficient topology optimization algorithm based on the function representation. *Applied Sciences*, 11(16):7409, 2021a

Co-author

1. Catherine Pakhomova, Dmitry Popov, Eugenii Maltsev, Iskander Akhatov, and Alexander Pasko. Software for bioprinting. *International Journal of Bioprinting*, 6(3), 2020

2. Evgenii Maltsev, Dmitry Popov, Svyatoslav Chugunov, Alexander Pasko, and Iskander Akhatov. An accelerated slicing algorithm for frep models. *Applied Sciences*, 11(15):6767, 2021

3. Alexander Safonov, Evgenii Maltsev, Svyatoslav Chugunov, Andrey Tikhonov, Stepan Konev, Stanislav Evlashin, Dmitry Popov, Alexander Pasko, and Iskander Akhatov. Design and fabrication of complex-shaped ceramic bone implants via 3d printing based on laser stereolithography. *Applied Sciences*, 10(20):7138, 2020

Conferences

Main author

1. Dmitry Popov, Evgenii Maltsev, Alexander Pasko, and Iskander Akhatov. Op-

timization of a shape and topology optimization for frep models. In *All-Russian Congress on the fundamental problems of theoretical and applied mechanics*, 2019

2. Dmitry Popov, Yulia Kuzminova, Evgenii Maltsev, Stanislav Evlashin, Alexander Safonov, Iskander Akhatov, and Alexander Pasko. Structural optimization of frep models and their additive manufacturing. In *CAASE20: The Conference on Advancing Analysis & Simulation in Engineering*, 2020a

3. Dmitry Popov, Galkin Timofey, Evgenii Maltsev, and Alexander Pasko. Web CAD/CAM system for additive manufacturing based on function representation. In *Beam technologies & laser applications*, 2021b

Co-author

1. Evgenii Maltsev, Dmitry Popov, Svyatoslav Chugunov, Alexander Pasko, and Iskander Akhatov. Using function representation of 3d models in additive manufacturing. In *All-Russian Congress on the fundamental problems of theoretical and applied mechanics*, 2019

Dedicated to my wife, who supported me during my PhD, and to my family, who instilled a love of knowledge in me

Contents

List of Symbols, Abbreviations

AA Affine Arithmetic. 86–90, 95, 96

AABB Axis-Aligned Bounding Boxes. 98, 99

AI Artificial Intelligence. 15

AM Additive Manufacturing. 7, 14–17, 33, 37, 48, 78, 79, 83, 115–117

API Application Programming Interface. 37

BRep Boundary Representation. 15, 16, 19–21, 34, 47

CAD Computer Aided Design. 9, 12, 15–17, 19–21, 33, 34, 36, 37, 39, 40, 115, 116

CAE Computer Aided Engineering. 36, 38, 39

CAM Computer Aided Manufacturing. 9, 16, 17, 26, 33, 34, 36–39, 78, 115, 116

CNC Computer Numerical Control. 15, 33, 37, 78, 101, 116, 117

CPU Central Processing Unit. 15, 106

CSG Constructive Solid Geometry. 15, 16, 19, 21

DLP Digital Light Processing. 11, 37, 38, 96, 112–114

DMD Direct Metal Deposition. 96, 112

DMT Direct Metal Tooling. 11, 112, 113

FDM Fused Deposition Modelling. 11, 96, 112, 113

FEM Finite Element Method. 18, 29, 34, 38, 51, 67, 70, 75, 76

FRep Function Representation. 7, 9–11, 15, 17–19, 21–24, 26, 27, 33–117

GPU Graphics Processing Unit. 15, 16, 42, 46, 115

IA Interval Arithmetic. 84–86, 89, 90, 94, 95

JSON JavaScript Object Notation. 37

OOP Object-Oriented Programming. 18

SIMP Solid Isotropic Material with Penalization. 15, 16

List of Figures

List of Tables

Chapter 1

Introduction

Additive Manufacturing (AM) is a perspective manufacturing technology with a 20% estimated compound annual growth rate from 2019 to 2025 [Altıparmak and Xiao, 2021]. This technology attracts the attention of industries because it provides lightweight products, low volume production, high design complexity, the ability to change designs frequently as needed [Abdulhameed et al., 2019], supply chain management for mass customization, and promising healthcare opportunities for bioprinting applications. The aerospace, car industry, medicine, architecture, and jewelry use AM. Several trends of AM development appeared in these industries. One of the developing directions of AM development is the design of metamaterials [Palani and Kannan, 2022]. This term defines different types of inner structures. Their design problem appears in many applications, including thermoelectrics [Zhang et al., 2022a], implants, and scaffolds [Sezer et al., 2021]. Moreover, researchers want to make a design more automated. It leads to the development of structural optimization in architecture [Liu et al., 2022b] and topology optimization in general [Liu et al., 2018].

AM breathed new life into topology optimization. Topology optimization was not attractive for conventional manufacturing processes because it required manual verification of obtained designs and their remodeling [Meng et al., 2020]. The limitations of these manufacturing processes restrict topology optimization algorithms from well physical implementation. AM provides more design freedom. Thus, many automatically generated designs are appropriate for this technology. However, there are still some challenges for topology optimization used for AM. The optimization algorithm can result in parts violating the resolution of a 3D printer or its acceptable overhang angle [Prathyusha and Babu, 2022]. The first option to solve these problems is to incorporate these constraints into the topology optimization algorithm. The second is to include support structures in the design process [Amir and Amir, 2021]. This way introduces a new branch of modeling tasks, i.e., the modeling of lattice structures. They can work as supports for overhang regions of parts. One can also use lattices for lightweight by filling them into inner volumes of parts. These lattice structures are one more subject of topology optimization. Topology optimization algorithms have more fundamental challenges except for mentioned ones. One of them is the performance of large-scale applications [Mukherjee et al., 2021]. It seems that Central Processing Unit (CPU) and Graphics Processing Unit (GPU) parallel computing [Liu et al., 2022a], [Duarte et al., 2015], [Xia et al., 2017] and Artificial Intelligence (AI) methods [Maksum et al., 2022] can solve this problem. One more challenge of topology optimization algorithms is that it still requires remodeling the part [Muzzupappa et al., 2010], [Bhosale and Sapkal, 2021]. The reason is different geometry representations for the design, optimization, and generation of Computer Numerical Control (CNC) programs for AM. The most widely used topology optimization method, Solid Isotropic Material with Penalization (SIMP) [Sigmund, 2001], uses voxel representation. Computer Aided Design (CAD) systems work with the hybrid geometric model. It uses Constructive Solid Geometry (CSG) [Requicha and Voelcker, 1977] and Boundary Representation (BRep) [Stroud, 2006]. Finally, the ".STL" format is the de facto standard for CNC generation software used for AM.

This thesis proposes an architecture of a modeling system for AM with efficient rendering, slicing, topology, and parameter optimization algorithms based on the unique geometry representation, Function Representation (FRep). All mentioned components of the system are crucial. The rendering algorithm provides drawing facilities that show a preview of the modeled object. In the scope of this thesis, slicing means procedures for generating CNC programs required for AM. Topology and parameter optimization components simplify the design process through automatic design suggestions for an arbitrary functional problem.

The main goal of the proposed system is to exclude geometry transformations for performing different tasks and preserving the required precision of models. It is not true for most existing CAD/CAM systems supporting topology optimization. Many of them use SIMP for this purpose [Schramm and Zhou, 2006], [Wang et al., 2019], [Vlah et al., 2020]. Therefore, optimized parts require transformation from voxels to an internal system representation.

Table 1.1 shows popular CAD products and the geometry representations they used. One can find a more detailed analysis of the listed and some other products in [Pakhomova et al., 2020]. Most of them work with BRep, and some extend it with hybrid representation techniques.

CAD systems widely utilize BRep despite its drawbacks from AM perspective [Oropallo and Piegl, 2016]. The leading position of this representation comes from CAD history and specific GPU interfaces. The Sketchpad system [Sutherland, 1964] proposed in 1963 considers lines and circles as key elements defining a solid. Thus, it exploited the idea of boundaries in 2D. Later 3D modeling system, UNIURF CAD [Bézier, 2014], proposed by Pierre Bezier in 1968, similarly uses surfaces as a tool to introduce a 3D volume. On the other hand, programming interfaces like OpenGL [Shreiner et al., 2009] provide shading languages based on boundary representation for efficient GPU usage.

Two critical problems of BRep and CAD systems [Piegl, 2005] concerning AM are model sizes and accuracy [Oropallo and Piegl, 2016]. The relatively compact representation used for visualization transforms into huge ".STL" files for 3D printing. It is a more sensitive issue for complex shapes with heterogeneous infill structures. That is why some systems chose hybrid geometry representation.

Many systems with hybrid representations operate with CSG trees [Requicha and Voelcker, 1977] proposed in 1977 for objects built of simple primitives by applying set-theoretic operations. However, more complex modeling tools, e.g., blending, require converting CSG models to BRep. Moreover, one needs to convert voxel models generated by SIMP into one of these representations for assembly modeling. These transformations from one representation to another cause issues with accuracy

[Piegl, 2005].

This thesis shows that the younger modeling approach based on FRep (see Section 2.1) can provide a CAD/CAM framework with a unique representation. FRep provides a compact symbolic representation of solids suitable for arbitrary rendering or manufacturing precision. FRep can become a new trend in the development of AM and CAD industries. According to Table 1.1, nTopology nTopology, Inc. [2022] is the only commercial system that operates with geometry in FRep. Some systems support it via plugins, e.g., Robert McNeel & Associates [2022], or enthusiastic research platforms, e.g., HyperFun Team [2022]. However, the novelty of FRep as a modeling technique explains its modest position in the market.

The main research objective is to provide a prototype of the FRep-based CAD/CAM system. This task consists of several sub-tasks. There are developing the FRep geometric core of the modeling system, providing an efficient rendering algorithm, developing the slicing routines for direct AM of FRep objects, and developing optimization algorithms for topology and parameter optimization of the FRep. The solution to these goals allows us to check the following hypotheses.

The principal hypothesis is if FRep can be the geometry representation that responds to all modeling and manufacturing requirements. Moreover, the thesis studies if the performance of rendering and slicing algorithms for FRep is comparable with classical methods. Finally, two hypotheses about optimization algorithms are if they are time efficient and have efficient predictive power concerning displacements under applied loads.

The rest of the thesis is a coherent manuscript that studies the previously mentioned research questions. It starts from the research background in Chapter 2 which discusses the current state of theories and concepts used in the research. Chapter 3 provides a more detailed description of thesis objectives and their novelty. Starting from Chapter 4, the FRep-based modeling system, the thesis delivers the results obtained in our research. Section 4.2, Section 4.3, Chapter 6, and Chapter 5 describe the components of the developed system. Section 5.8 tells about the algorithm for multimaterial topology optimization. The last Chapter 7 provides a summary of the conducted research.

This thesis considers used FRep, the level set method, and the Finite Element Method (FEM) with an ersatz material model in the theoretical part of the research. Implementation of the proposed algorithms mainly relies on Object-Oriented Programming (OOP) [Stroustrup, 1988] and parallel computing. Let us overview these concepts in the following chapter.

#	CAD name	Comment on geometry representation
1	Autodesk AutoCAD Autodesk Inc. [2022a]	AutoCAD has closed standard for the internal geometry representation but we can suppose that it is based on the BRep because of its modeling tools
2	Dassault Systèmes SOLIDWORKS Dassault Systèmes [2022c]	SOLIDWORKS documentation says that the system uses reference geometry but its description is similar to BRep
3	Dassault Systèmes CATIA Dassault Systèmes [2022a]	CATIA documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
4	PTC Creo Parametric PTC [2022a]	We can read about BRep used in Creo Parametric in its documentation
5	FreeCAD The FreeCAD Team [2022]	FreeCAD is the open source project and the information about BRep used as its geometry representation is available in the documentation
6	Autodesk Inventor Autodesk Inc. [2022c]	The documentation of Autodesk Inventor reefers to BRep used in Autodesk AutoCAD
7	Siemens NX Siemens [2022b]	The geometry representation of Siemens NX is not a public information but we can assume that Siemens NX uses both BRep and CSG from the Shih's book [Shih, 2022]
8	Autodesk Fusion 360 Autodesk Inc. [2022e]	The documentation of Autodesk Fusion 360 says that it uses BRep
9	Rhinoceros Rhino Robert McNeel & Associates [2022]	Rhino uses a geometric kernel based on BRep but it has plugins whose can operate with other kind of representations: CSG, voxels and FRep
10	PTC Onshape PTC [2022b]	There is no available information about geometry representation used in PTC Onshape. However, we can suppose that this web-based CAD system inherited BRep from its desktop ancestor, PTC Creo Parametric
11	Siemens Solid Edge Siemens [2022a]	Public documentation of Siemens Solid Edge describes its geometry representation, i.e. BRep
12	Bentley Systems MicroStation Bentley Systems Inc. [2022]	The documentation of Bentley Systems MicroStation says that it works with BRep and has utilities to operate with voxel representation
13	OpenSCAD Marius Kintel [2022]	OpenSCAD is an open source project. It uses CSG and BRep according its documentation
14	Autodesk Revit Autodesk Inc. [2022d]	Revit documentation says that it uses BRep

Table 1.1: Geometry representations in CAD systems.

15	BricsCAD Bricsys NV [2022]	BricsCAD documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
16	LibreCAD The LibreCAD Team [2022]	LibreCAD is an open source project. It uses BRep according its documentation
17	IMSI Design TurboCAD IMSI Design LLC [2022]	TurboCAD documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
18	nanoCAD Nanosoft [2022]	nanoCAD documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
19	IntelliCAD IntelliCAD Technology Consortium [2022]	IntelliCAD documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
20	IRONCAD IronCAD, LLC [2022]	IRONCAD documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
21	Dassault Systèmes DraftSight Dassault Systèmes [2022b]	DraftSight documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools. This system has a powerful tools for rather 2D than 3D modelling
22	Alibre Design Alibre, LLC [2022]	DraftSight documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
23	Graphisoft Archicad Graphisoft [2022]	Archicad documentation says that it uses BRep
24	progeCAD progeSOFT [2022]	DraftSight documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
25	Gstarsoft GstarCAD Gstarsoft Co.,Ltd [2022]	GstarCAD documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
26	TrueCAD Jytra Technology Solutions [2022]	GstarCAD documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools

Table 1.1: Geometry representations in CAD systems (Continued).

27	ZWCAD ZWSOFT CO., LTD. [2022]	ZWCAD documentation says that it uses BRep
28	cadwork cadwork Software [2022]	cadwork documentation says that it uses both BRep and CSG
29	SolveSpace SolveSpace contributors [2022]	SolveSpace documentation says that it uses BRep
30	Ansys SpaceClaim ANSYS, Inc [2022]	SpaceClaim uses BRep and finite elements similar to voxels for simulations
31	CorelCAD Corel Corporation [2022]	GstarCAD documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
32	Top Systems T-FLEX CAD Top Systems [2022]	T-FLEX CAD documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
33	LeoCAD LeoCAD.org [2022]	LeoCAD is an open source CAD system for 3D modeling with LEGO bricks
34	Vectorworks Vectorworks, Inc. [2022]	Vectorworks documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
35	Autodesk AutoCAD Architecture Autodesk Inc. [2022b]	AutoCAD Architecture documentation does not provide a public information about its internal geometry representation but it is likely BRep and CSG with primitives suitable for architecture
37	Vizerra Revizto Vizerra SA [2022]	Revizto documentation does not provide a public information about its internal geometry representation but it is likely BRep because of its modeling tools
38	BRL-CAD Army Research Laboratory [2022]	Open-source software by United States Army Research Laboratory. Its documentation says the it uses both BRep and CSG
39	VariCAD VariCAD [2022]	VariCAD documentation says that it uses BRep
40	nTopology nTopology, Inc. [2022]	nTopology blog says that it uses FRep, BRep, CSG and voxel representation

Table 1.1: Geometry representations in CAD systems (Continued).

"if I have seen further it is by standing on the shoulders of Giants."

Isaac Newton, 1675

Chapter 2

Background

2.1 Function representation

FRep is a technique of geometric modeling [Pasko et al., 1995], [Kambampati et al., 2021], [Tereshin et al., 2021], [Zhang et al., 2022b]. From a practical perspective, its ability to precisely describe two- and three-dimensional shapes is more attractive. However, this theory covers modeling sets in any finite dimension. The three key concepts of FRep are objects, operations, and relations.

2.1.1 Objects

Objects in FRep are closed subsets of the n-dimensional Euclidean space $\mathbb{E}^n$. They are defined through inequalities:

$$f(x_1, ..., x_n) \geq 0, \tag{2.1}$$

where $f(x_1, ..., x_n)$ is a real continuous function defined in $\mathbb{E}^n$. This function is named a defining function.

The defining function introduces a ternary classification for all points in the Euclidean space $\mathbb{E}^n$. The function is positive in the open set of the interior of the object. The boundary of the object is the zero-level set of f. The function is negative in the open set of the exterior of the object.

This property of the defining function allows us to use it for interactions with shapes. First, one can create rendering algorithms to draw objects with arbitrary precision. Second, it is possible to slice objects for their 3D printing [Song et al., 2018]. Finally, there are introduced operations and relations on objects.

2.1.2 Operations

FRep is an extension to "implicit" models, which are based on equations of new primitives. FRep exploits the tree structure produced by applying of operations. Tracing of this structure generates the function value at the point.

Since the defining function introduces the ternary classification of the space, set-theoretic operations correspond to a three-valued logic. Research in this area led to the theory of R-functions [Rvachev, 1982], [Rvachev et al., 2001], [Kravchenko et al., 2020], [Kravchenko et al., 2021].

Table 2.1 contains a list of possible some R-functions used for set-theoretic union, intersection, and subtraction:

Group	Operation	Equation
$\mathfrak{R}_\alpha$	Union	$\frac{1}{1+\alpha}\left(f_1 + f_2 + \sqrt{f_1^2 + f_2^2 - 2\alpha f_1 f_2}\right)$
	Intersection	$\frac{1}{1+\alpha}\left(f_1 + f_2 - \sqrt{f_1^2 + f_2^2 - 2\alpha f_1 f_2}\right)$
	Subtraction	$\frac{1}{1+\alpha}\left(f_1 - f_2 - \sqrt{f_1^2 + f_2^2 + 2\alpha f_1 f_2}\right)$
$\mathfrak{R}_0^m$	Union	$\left(f_1 + f_2 + \sqrt{f_1^2 + f_2^2}\right)(f_1^2 + f_2^2)^{\frac{m}{2}}$
	Intersection	$\left(f_1 + f_2 - \sqrt{f_1^2 + f_2^2})(f_1^2 + f_2^2\right)^{\frac{m}{2}}$
	Subtraction	$\left(f_1 - f_2 - \sqrt{f_1^2 + f_2^2}\right)(f_1^2 + f_2^2)^{\frac{m}{2}}$
$\mathfrak{R}_p$	Union	$f_1 + f_2 + (\lvert f_1\rvert^p + \lvert f_2\rvert^p)^{\frac{1}{p}}$
	Intersection	$f_1 + f_2 - (\lvert f_1\rvert^p + \lvert f_2\rvert^p)^{\frac{1}{p}}$
	Subtraction	$f_1 - f_2 - (\lvert f_1\rvert^p + \lvert f_2\rvert^p)^{\frac{1}{p}}$
$\mathfrak{R}_C$	Union	$\left(\frac{f_1+f_2}{2}\right)^m sign(f_1 + f_2)^{m+1} + \left(\frac{f_1-f_2}{2}\right)^m sign(f_1 - f_2)^m$
	Intersection	$\left(\frac{f_1+f_2}{2}\right)^m sign(f_1 + f_2)^{m+1} - \left(\frac{f_1-f_2}{2}\right)^m sign(f_1 - f_2)^m$
	Subtraction	$\left(\frac{f_1-f_2}{2}\right)^m sign(f_1 - f_2)^{m+1} - \left(\frac{f_1+f_2}{2}\right)^m sign(f_1 + f_2)^m$

Table 2.1: Equations for set-theoretic operations.

The equations in Table 2.1 have three parameters. The first is α. It is a function such that $-1 < \alpha(f_1, f_2) \leq 1$. Two more parameters are the constants $m > 0$ and $p > 1$.

The most widely used group of set-theoretic operations is $\mathfrak{R}_\alpha$. The researchers exploit the equations of this group with $\alpha \equiv 0$ and $\alpha \equiv 1$. The last value of $\alpha \equiv 1$ led to the following set of equations:

$$\begin{aligned} f_1 \vee_0 f_2 &= max(f_1, f_2), \\ f_1 \wedge_0 f_2 &= min(f_1, f_2), \\ f_1 \setminus_0 f_2 &= min(f_1, -f_2). \end{aligned}$$

Set-theoretic operations belong to more broad class of binary operations, but unary operation exist in FRep as well. The simplest example of such an operation is the negation $\mathbb{E}^n \setminus \Omega$ of a subset Ω. In terms of FRep, it is $-f$, where $f = f(x_1, ..., x_n)$ is the defining function of Ω.

However, most unary operations in FRep are space mapping operations [Pasko et al., 1999], [Schmitt et al., 1999], [Savchenko et al., 1995], [Fryazinov et al., 2013]. This means that the defining function is still the same, but some mapping $\mathbb{E}^n \to \mathbb{E}^n$ is applied to the points of the initial Euclidean space. Table 2.2 shows a list of commonly used space mapping operations:

Operation	Equation	Parameters
Scaling	$\left(\frac{x_1}{s_1}, ..., \frac{x_n}{s_n}\right)$	$s_1, ..., s_n$ are scaling factors along corresponding directions
Translation	$(x_1 - x_1^0, ..., x_n - x_n^0)$	$(x_1^0, ..., x_n^0)$ is the new origin point of the object
Rotation (n=2)	$(x_1 \cos\theta + x_2 \sin\theta,$ $-x_1 \sin\theta + x_2 \cos\theta)$	θ is a rotation angle
Global tapering	$\left(\frac{x_1}{s_1(x_n)}, ..., \frac{x_{n-1}}{s_{n-1}(x_n)}, x_n\right)$	$s_1(x_n), ..., s_{n-1}(x_n)$ are scaling factors along corresponding directions, those depend on the x_n
Global axis twist (n=3)	$(x_1 \cos\theta + x_2 \sin\theta,$ $-x_1 \sin\theta + x_2 \cos\theta,$ $x_3)$	$\theta = \theta(x_3)$ is a function depending on x_3, that defines rotation angle. It shall be continuous for a smooth twist

Table 2.2: Equations for space mapping operations.

One more wide-spread operation is blending [Hsu, 2018], [Bhooshan et al., 2018], [Corker-Marin et al., 2018], [You, 2022], [Tereshin et al., 2020]. It can be used with set-theoretic union, intersection, or subtraction:

$$
\begin{gathered}
f_1 \vee f_2 + d(f_1, f_2), \\
f_1 \wedge f_2 + d(f_1, f_2), \\
f_1 \setminus f_2 + d(f_1, f_2),
\end{gathered}
$$

where $d(f_1, f_2)$ in the displacement introduced by the blending operation:

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

where a_0, a_1, and a_2 are constants. The sign of the parameter a_0 controls whether the material will be added or removed from the result object. The parameters $a_1 > 0$ and $a_2 > 0$ are responsible for the amount of material added or removed around each of the initial objects.

One more exotic, but quite useful operation is bounded blending [Pasko et al., 2005]. This is an example of a ternary operation. The bounded blending has the same form as simple blending, but with more complicated displacement function $d = d(f_1, f_2, f_3)$:

$$
\begin{aligned}
d(r) &= \begin{cases} \frac{\left(1-r^2\right)^3}{1+r^2}, r < 1 \\ 0, r \geq 1 \end{cases}, \\
r &= \frac{r_1^2}{r_1^1 + r_2^2}, \\
r_1^2(f_1, f_2) &= \left(\frac{f_1}{a_1}\right)^2 + \left(\frac{f_2}{a_2}\right)^2, \\
r_2^2(f_3) &= \begin{cases} \left(\frac{f_3}{a_3}\right)^2, f_3 > 0 \\ 0, f_3 \leq 0 \end{cases}.
\end{aligned}
$$

This operation uses the object defined by f_3 to constrain the added or removed material with its volume, and the new parameter a_3 more sensitively controls the amount of material changes.

All the above mentioned examples are not a complete set of possible operations of FRep. New operations can appear in specific modeling applications. The main requirement to the operation is that it generates new continuous function

2.1.3 Relations

The last key concept of FRep is relations [Tereshin et al., 2021]. They serve to verify the hypothesis about the modeled objects. The mathematical formalization of relations is predicates. Let us consider three useful examples.

The fist one is inclusion relation. This relation checks the hypothesis to see whether the object Ω_1 is included in the object Ω_2 or not. It can be written as a bi-valued predicate using the respective defining functions f_1 and f_2:

$$P_1(\Omega_1, \Omega_2) = \begin{cases} 0, \text{ if } \exists (x_1, ..., x_n) : f_2(x_1, ..., x_n) < 0 \text{ and } f_1(x_1, ..., x_n) \geq 0, \\ 1, \text{ if } f_2(x_1, ..., x_n) \geq 0 \; \forall (x_1, ..., x_n) : f_1(x_1, ..., x_n) \geq 0. \end{cases}$$

One more example of relations is the point membership:

$$P_2((x_1, ..., x_n), \Omega) = \begin{cases} 0, \text{ if } f(x_1, ..., x_n) < 0, \\ 1, \text{ if } f(x_1, ..., x_n) = 0, \\ 2, \text{ if } f(x_1, ..., x_n) > 0, \end{cases}$$

where $f(x_1, ..., x_n)$ is a defining function of object Ω. Simple rendering algorithms can exploit this relation to draw objects.

The next relation can be used for collision detection. Collision detection is an important procedure for CAM systems and game development. The intersection relation can help with construction of a collision detection algorithm:

$$P_1(\Omega_1, \Omega_2) = \begin{cases} 0, \text{ if } f_1(x_1, ..., x_n) \wedge f_2(x_1, ..., x_n) < 0 \; \forall (x_1, ..., x_n) \in \mathbb{E}^n, \\ 1, \text{ if } \exists (x_1, ..., x_n) : f_1(x_1, ..., x_n) \wedge f_2(x_1, ..., x_n) \geq 0, \end{cases}$$

where $\wedge$ is an FRep operation for the set-theoretic intersection, e.g., from Table 2.1.

FRep is a technique used in different fields. Particularly, it found applications in level set methods.

2.2 Topology Optimization Based on Level Set Methods

Topology optimization algorithms based on level set methods [Osher and Sethian, 1988], [Fedkiw and Osher, 2002] exploit the concept of FRep defining function. These methods consider continuous and differentiable defining functions $f(x_1, ...x_n)$ with $n = 2$ or $n = 3$. Moreover, these functions depend on the pseudo-time variable, i.e., $f = f(t, x_1(t), ..., x_n(t))$. This variable introduces continuous changes in the defining function during the optimization process. Optimization algorithms use discrete values of increasing pseudo-time iteratively.

The descent series of a topology optimization algorithm based on level set methods consists of functions $f(t_i, x_1(t_i), ..., x_n(t_i))$. These values come from the solution of the Hamilton-Jacobi-like partial differential equation:

$$\frac{\partial f}{\partial t} = V|\boldsymbol{\nabla} f|, \tag{2.2}$$

where V is a velocity orthogonal to the boundary of the object and t_i are samples increased in time, usually defined by a numerical solution algorithm [Wang et al., 2003], [Allaire et al., 2004], [Burger et al., 2004].

The specific optimization problem defines the magnitude of velocity V. There are dynamic optimization problems for stiffness ([Wang et al., 2007], [Takezawa et al., 2010], [Wei et al., 2010], [Vogiatzis et al., 2017]), frequency response ([Shu et al., 2011]), thermal ([Yaji et al., 2015], [Xia et al., 2018], [Yu et al., 2019]) and fluids ([Amstutz and Andrä, 2006], [Zhou and Li, 2008], [Challis and Guest, 2009]). However, this thesis focuses on the compliance minimization problem in current research.

The following notation is used to introduce the problem statement. The displacement of a point $(x_1, ..., x_n)$ is $\boldsymbol{u}$; E_{ijkl} is the Hook elasticity tensor; $\varepsilon_{ij}(\boldsymbol{u})$ and $\varepsilon_{kl}(\boldsymbol{u})$ are linearized strain tensors:

$$\boldsymbol{u} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}, \varepsilon_{xx} = \frac{\partial u_x}{\partial x}, \varepsilon_{yy} = \frac{\partial u_y}{\partial y}, \varepsilon_{zz} = \frac{\partial u_z}{\partial z},$$

$$\varepsilon_{xy} = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}, \varepsilon_{xz} = \frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}, \varepsilon_{yz} = \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y},$$

There are several options for the optimization objective functions. Two possible choices are listed below:

$$J_1(\Omega) = \int_\Omega E_{ijkl}\varepsilon_{ij}(\boldsymbol{u})\varepsilon_{kl}(\boldsymbol{u})d\Omega,$$
$$J_2(\Omega) = \left(\int_\Omega w(x_1, ..., x_n)|\boldsymbol{u} - \boldsymbol{u_{obj}}|^\alpha d\Omega\right)^{\frac{1}{\alpha}},$$

where $\boldsymbol{u_{obj}}$ is the target displacement, $\alpha \geq 2$ is a constant and $w(x_1, ..., x_n)$ is a given non-negative weighting factor.

Formally, the considered compliance minimization problem for an elastic body with mixed boundary conditions can be written as following:

$$\begin{aligned} \min_f \quad & \int_\Omega E_{ijkl}\varepsilon_{ij}(\boldsymbol{u})\varepsilon_{kl}(\boldsymbol{u})d\Omega, \\ \text{s.t.} \quad & a(\boldsymbol{u}, \boldsymbol{v}, f) = l(\boldsymbol{v}, f), \quad \forall \boldsymbol{v} \in U, \\ & \boldsymbol{u}|_{\Gamma_1} = \boldsymbol{u_0}, \\ & E_{ijkl}\varepsilon_{ij}(\boldsymbol{u})|_{\Gamma_2} = \boldsymbol{p}, \\ & \int_\Omega d\Omega \leq V_0, \end{aligned} \tag{2.3}$$

where

$$a(\boldsymbol{u}, \boldsymbol{v}, f) = \int_\Omega E_{ijkl}\varepsilon_{ij}(\boldsymbol{u})\varepsilon_{kl}(\boldsymbol{v})d\Omega,$$
$$l(\boldsymbol{v}, f) = \int_\Omega \boldsymbol{gv} + \nabla(\boldsymbol{pv}n)d\Omega,$$

$\boldsymbol{u_0}$, $\boldsymbol{p}$ and $\boldsymbol{g}$ are the constant vectors that represent the displacement, traction, and body force given, respectively; $a(\boldsymbol{u}, \boldsymbol{v}, f)$ is the linear elastic equilibrium equation written in its weak variational form and $l(\boldsymbol{v}, f)$ is the linear form of the load, with $\boldsymbol{v}$ denotes a virtual displacement field in the space U of kinematically admissible displacement fields; V_0 is the maximum allowable volume.

It is not the only way for introducing the topology optimization problem. Some papers work with the dual problem where the volume $\int_\Omega d\Omega$ is an objective functions. Optimization algorithms can vary for different forms of the problem statements. The proposed level set method used in this research is described in Chapter 5.

2.3 Finite element method with ersatz material model

FEM is a powerful tool used in many applications, but this thesis focuses on its particular formulation. You can find a comprehensive description of the method in the existing literature, e.g. [Zienkiewicz et al., 2005], [Dhatt et al., 2012], [Rao, 2017], and [Reddy, 2019]. However, this thesis discusses only its application for elastic solids with isotropic material model.

FEM simulates mechanical properties of solids. It exploits the Hooke's law:

$$\boldsymbol{\sigma} = \boldsymbol{c\varepsilon},$$

where

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{yz} \\ \sigma_{xz} \\ \sigma_{xy} \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{yz} \\ \varepsilon_{xz} \\ \varepsilon_{xy} \end{pmatrix},$$

$$\boldsymbol{c} = \begin{pmatrix} \frac{E(1-\nu)}{(1-2\nu)(1+\nu)} & \frac{E\nu}{(1-2\nu)(1+\nu)} & \frac{E\nu}{(1-2\nu)(1+\nu)} & 0 & 0 & 0 \\ \frac{E\nu}{(1-2\nu)(1+\nu)} & \frac{E(1-\nu)}{(1-2\nu)(1+\nu)} & \frac{E\nu}{(1-2\nu)(1+\nu)} & 0 & 0 & 0 \\ \frac{E\nu}{(1-2\nu)(1+\nu)} & \frac{E\nu}{(1-2\nu)(1+\nu)} & \frac{E(1-\nu)}{(1-2\nu)(1+\nu)} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{E\nu}{2(1+\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{E\nu}{2(1+\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{E\nu}{2(1+\nu)} \end{pmatrix}$$

and $\boldsymbol{\sigma}$ is a vector of stress components at an elementary volume as shown in Figure 2-1(a); $\boldsymbol{\varepsilon}$ is a vector of corresponding strain components; E is a Young's modulus and ν is a Poison's ratio.

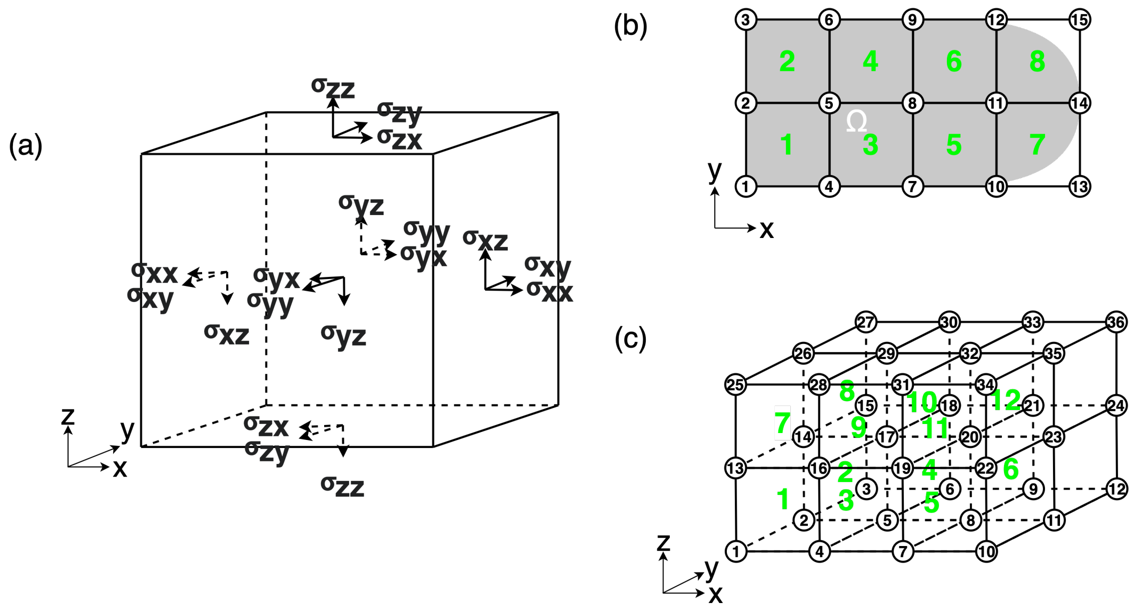


Figure 2-1: Components of the mechanical simulation scheme: (a) elementary cubic volume with stress components; (b) square finite element grid placed over the solid Ω with numbered nodes and elements; (c) cubic finite element grid with numbered nodes and elements.

Uniform square (Figure 2-1(b)) and cubic (Figure 2-1(c)) finite elements approximate the solution of a mechanical problem. Green numbers mark elements, and numbered circles show nodes in Figure 2-1. Bilinear and trilinear interpolations approximate components of the displacement vector $\boldsymbol{u}$ within an element in 2D and 3D, respectively.

The plane stress model is chosen for simulation in 2D. Thus, the Hook elasticity tensor E_{ijkl} used in Equation 2.3 for 2D case has the following form:

$$E_{ijkl} = \frac{E}{1-\nu^2}\begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{pmatrix}.$$

The element stiffness matrix k_e for square grid for this elasticity tensor E_{ijkl} is introduced as following for the first element in Figure 2-1(b):

$$k_e = \int_{V_e} B^T E_{ijkl} B dV, \tag{2.4}$$

where

$$B = \begin{pmatrix} \frac{\partial}{\partial\xi} & 0 \\ 0 & \frac{\partial}{\partial\eta} \\ \frac{\partial}{\partial\xi} & \frac{\partial}{\partial\eta} \end{pmatrix} \begin{pmatrix} N_1 & 0 & N_4 & 0 & N_5 & 0 & N_2 & 0 \\ 0 & N_1 & 0 & N_4 & 0 & N_5 & 0 & N_2 \end{pmatrix},$$

$$\begin{aligned} N_1 &= \frac{1}{4}(1-\xi)(1-\eta), \\ N_4 &= \frac{1}{4}(1+\xi)(1-\eta), \\ N_5 &= \frac{1}{4}(1+\xi)(1+\eta), \\ N_2 &= \frac{1}{4}(1-\xi)(1+\eta). \end{aligned}$$

N_1, N_4, N_5 and N_2 are shape functions for displacement approximation within this element. Subscript numbers refer to corresponding nodes. Equations for the shape functions are provided in local coordinate system ξ, η with origin in the center of cubic element and its boundaries located at $\xi = \pm 1, \eta = \pm 1$. Thus, the displacement approximation $\boldsymbol{u}$ within the first element is:

$$\boldsymbol{u} = \begin{pmatrix} \boldsymbol{u_1} & \boldsymbol{u_4} & \boldsymbol{u_5} & \boldsymbol{u_2} \end{pmatrix} \begin{pmatrix} N_1 \\ N_4 \\ N_5 \\ N_2 \end{pmatrix}$$

where $\boldsymbol{u_1}, \boldsymbol{u_4}, \boldsymbol{u_5}$ and $\boldsymbol{u_2}$ are displacement vectors at the corresponding nodes.

The element stiffness matrix Equation 2.4 is the same for all uniform square elements as in Figure 2-1(b). It can be explicitly calculated by symbolic integration of the Equation 2.4. Special software, such as The MathWorks, Inc. [2022], can perform the integration and the result matrix will be the same as one used in the classical topology optimization paper [Sigmund, 2001].

The use of the ersatz material model allows to avoid the re-meshing of solids during optimization. It means varying Young's modulus of the material instead of deforming the square elements. For example, the exact value of the material Young's modulus is used for elements $1, 2, 3, 4, 5$, and 6 in Figure 2-1(b), but it is reduced proportionally to the volume occupied by the solid body Ω for elements 7 and 8. The Young's modulus of stainless steel used for experimental validation is supposed to be 205 GPa, and the Poisson's ratio, ν, is assumed to be 0.27 (see Section 5.2). The properties of the material from our laboratory were studied in [Kuzminova et al., 2019].

Chapter 3

Thesis Objectives And Novelty

The objectives of this thesis rely on the identified drawbacks of the CAD/CAM and topology optimization software for AM.

3.1 Goals

The main goal is to propose the system with a unique representation for modeling and optimization. This thesis considers FRep as a good candidate for this role. The proposed system should satisfy specific requirements for the CAD software applied to AM design. Two key points of such a special challenges are complexity of the designed shapes and their parametrization. Thus, the thesis studies if FRep can cope with there tasks.

FRep is not a panacea for all modeling and manufacturing tasks. Its implementation in CAD/CAM systems has several bottlenecks those should be investigated.

The first one is slow rendering facilities of this representation. Since functions define solids in FRep, their drawing requires significant number of their evaluations. It is a time consuming task when functions are complex. Therefore, this work needs to investigate existing rendering techniques exploiting parallel computations and propose an efficient drawing procedure.

The second important algorithm is slicing. Within this thesis, the term slicing denotes a procedure of CNC program generation for AM. However, FRep is not the only representation where slicing can be challenging. It implies the conversion of

3D models to so-called 2.5D geometry representation, which is challenging except you use 2.5D models, e.g., layered depth-normal images [Chen and Wang, 2011], as a geometric kernel. Nonetheless, the thesis needs to propose an efficient and competitive slicing algorithm.

The last issue connected with FRep efficiency is the implementation of mechanical simulation algorithms. The problem in this aspect has the same nature as two previously mentioned points. The classical FEM requires solid partition into elements which also relies on the boundary extraction. The thesis tries to avoid this step by using FEM with the ersatz material model.

This approach for simulations came from level set methods used in different applications. One more goal of the thesis is establishing more connections between FRep and level set methods to improve modeling, simulation and optimization facilities.

Finally, this thesis assume to realize direct manufacturing tools in the developed CAD/ CAM software.

3.2 Novelty

The proposed system has two essentially novel aspects.

The first one is the uniqueness of the geometry representation for all modeling and manufacturing tasks. You can find the proposed FRep based system for CAD, e.g. [Pasko et al., 1995], nTopology, Inc. [2022] and ephtracy [2021]. The are also simulation systems based on FRep with simple design facilities, e.g. [Shapiro and Tsukanov, 2002] or even older Soviet Ukraine system called "Pole" [Rvachev, 1982]. The paper [Song et al., 2018] proposes FRep-based CAM tools. However, there was no FRep-based system that aggregates all three components: CAD, simulations and CAM.

The second novel aspect of the proposed FRep system is its collaboration capabilities. The architecture proposed in Section 4.1 describes a web application with common space of objects and modeling tools exploiting users computational resources. There are CAD systems based on the BRep, e.g. PTC [2022b]. However, the current thesis is the first research, which delivers such a system based on FRep.

One more novel point of this research is the study of interconnections between FRep and level set methods. Chapter 5 delivers with results on control of optimization process via FRep modeling techniques.

Moreover, Section 5.8 proposes a new multimaterial topology optimization algorithm based on the model with dominant material.

Chapter 4

FRep-based modeling system

4.1 Architecture of the system

Figure 4-1 shows main components of the proposed CAD/CAM FRep-based system. It consists of three key parts: CAD component, CAM component and Computer Aided Engineering (CAE) component.

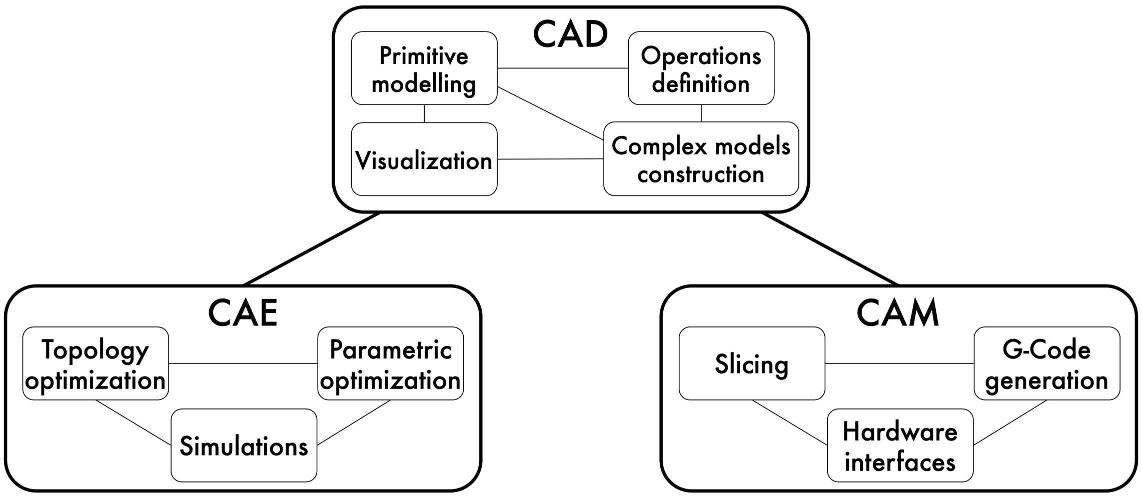


Figure 4-1: Main components of the CAD/CAM FRep-based system.

The CAD component includes tools for basic modeling activities. Primitive modeling, operation definition and complex models construction blocks make up FRep geometric gore of the modeling system. The primitive modeling tool realizes the concept of FRep objects. It stores defining functions of solids in symbolic form and its parameters, e.g. dimensions. The complex model construction tool realizes the processing of tree-like structures made of existing primitives. The operation definition tool introduces operations that can be used for constructing of these structures. The last part of these components is the visualization tool that supports the design process.

The CAD component was implemented with JavaScript, PHP, and GLSL languages. JavaScript code supports the main functionality of the system and user interactions with its interface. You can see its class diagram in Figure 4-2. The `Element` class and its children's classes `Primitive` and `Structure` provide fields and methods for working with solid objects. Their FRep defining functions are arranged as codes in JavaScript saved in the `jscode` field. The class `Structure` stores more complex objects made of several primitives in a JavaScript Object Notation (JSON) format ECMA [2022]. JSON is a lightweight data-interchange format. It is easy for machines to parse and generate. Our system uses this format for storing symbolic descriptions of primitives, a composition of primitives in structures, and attributes of models. Methods `loadFromCloud`, `saveToFromCloud`, and `removeFromCloud` realize interactions with MySQL database exploiting Lumen Taylor Otwell [2022] PHP framework. Visualization routines exploit WebGL The Khronos Group Inc [2022] Application Programming Interface (API) running shaders written in GLSL (see Section 4.3).

The CAM component is responsible for slicing, CNC program generation and interaction with 3D printing hardware. Since, AM is a layer by layer manufacturing technique, slicing means extraction of a layer from the modeled object. As soon as layers are extracted, the system must prepare the whole CNC program that defines manufacturing process. Usually, it is a G-Code file with predefined list of interpretative machine commands. Finally, CAM component of the system has to deliver the generated control program to the user or a 3D printer directly.

The implemented CAM component of the system is compatible with several types of AM machines available in Additive Manufacturing Laboratory of Skoltech Center For Materials Technologies. Routines of this component depend on the type of equipment used for 3D printing. It can prepare a G-Code for Wanhao Duplicator 7, Zortrax Inkspire and Liquid Crystal Pro Digital Light Processing

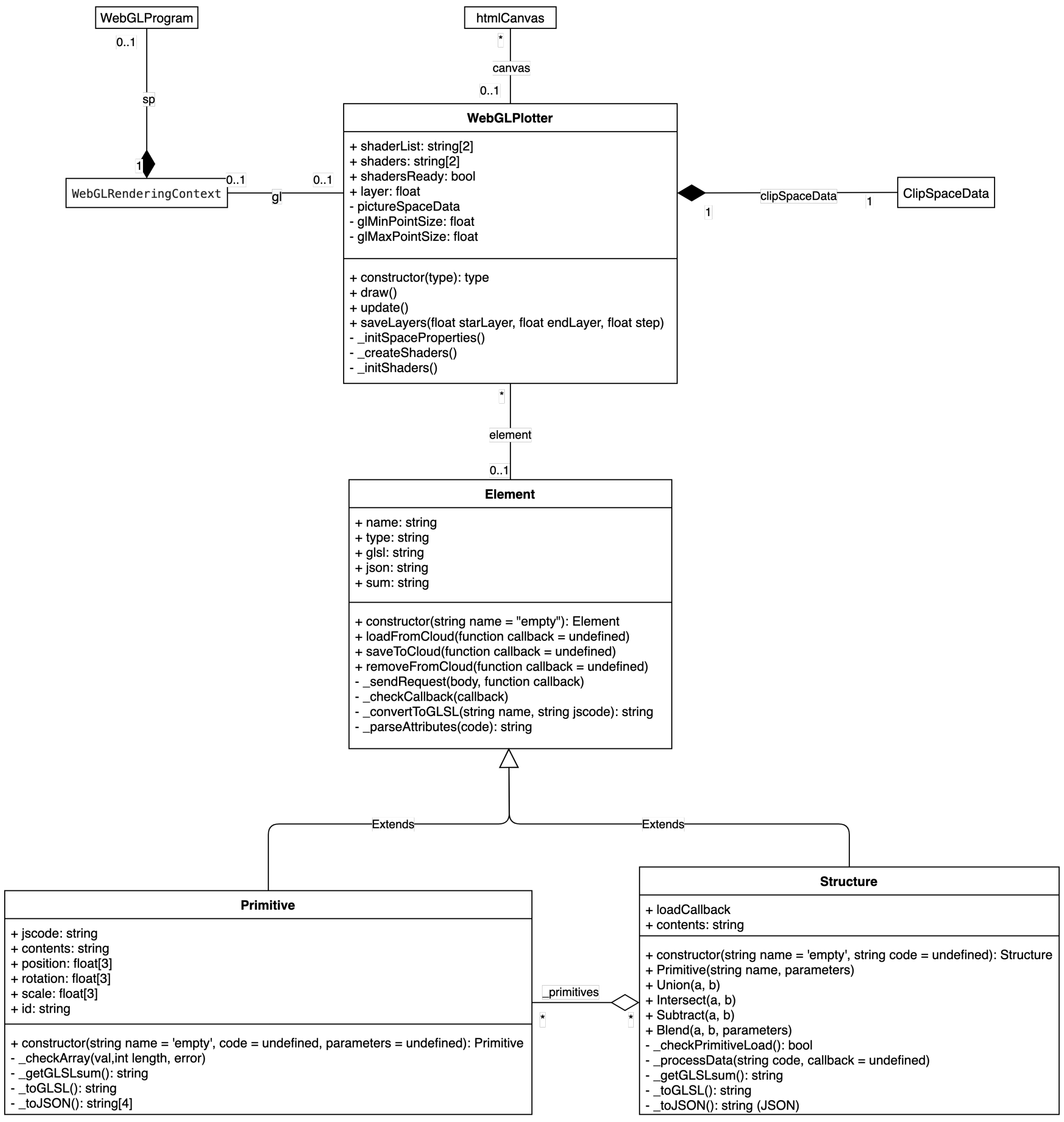

WebGLProgram
0..1
sp
1
WebGLRenderingContext
0..1
gl
0..1
htmlCanvas
*
canvas
0..1
WebGLPlotter
+ shaderList: string[2]
+ shaders: string[2]
+ shadersReady: bool
+ layer: float
- pictureSpaceData
- glMinPointSize: float
- glMaxPointSize: float
+ constructor(type): type
+ draw()
+ update()
+ saveLayers(float starLayer, float endLayer, float step)
- _initSpaceProperties()
- _createShaders()
- _initShaders()
1
clipSpaceData
1
ClipSpaceData
*
element
0..1
Element
+ name: string
+ type: string
+ glsl: string
+ json: string
+ sum: string
+ constructor(string name = "empty"): Element
+ loadFromCloud(function callback = undefined)
+ saveToCloud(function callback = undefined)
+ removeFromCloud(function callback = undefined)
- _sendRequest(body, function callback)
- _checkCallback(callback)
- _convertToGLSL(string name, string jscode): string
- _parseAttributes(code): string
Extends
Extends
Primitive
+ jscode: string
+ contents: string
+ position: float[3]
+ rotation: float[3]
+ scale: float[3]
+ id: string
+ constructor(string name = 'empty', code = undefined, parameters = undefined): Primitive
- _checkArray(val,int length, error)
- _getGLSLsum(): string
- _toGLSL(): string
- _toJSON(): string[4]
_primitives
*
*
Structure
+ loadCallback
+ contents: string
+ constructor(string name = 'empty', string code = undefined): Structure
+ Primitive(string name, parameters)
+ Union(a, b)
+ Intersect(a, b)
+ Subtract(a, b)
+ Blend(a, b, parameters)
- _checkPrimitiveLoad(): bool
- _processData(string code, callback = undefined)
- _getGLSLsum(): string
- _toGLSL(): string
- _toJSON(): string (JSON)

Figure 4-2: The class diagram of the frontend part of the system.

(DLP) printers exploiting `saveLayers` method of `WebGLPlotter` class. G-Code generation procedures is realized for Insstek MX-1000, Ultimaker S5 and Picaso DesignerXPro printers. However, implementation of CAM routines for these printers differ from DLP-oriented scheme (for more detailed discussion see Chapter 6).

The CAE component of the system performs topology and parametric optimization of solids based on mechanical simulations. Topology optimization and parametric optimization (see Chapter 5) algorithms solve the compliance minimization problem for given domain and loads. Mechanical simulations exploit FEM with ersatz material model (see Section 2.3).

Chapter 5 describes the realization of CAE component with references to MATLAB implementation. This component is a suite of programs for generation of an FRep defining function that can be processed by CAD and CAM components of the system.

Two options of the system implementation were considered. The form of a cloud-based service seems more preferable than a desktop application for several reasons. The first one is opportunities of collaboration provided by web-oriented realization. Users can have access to the shared models and work with them exploiting their local computational resources. The second reason of choosing the cloud-based solution is absence of the installation step. Finally, the thesis shows that the development of programming tools, e.g. The Khronos Group Inc [2022], for web-oriented programming is reached the point when a complete CAD/CAM system can be created. Following chapters provide more details on the implementation of the system.

4.2 FRep geometric core of the modeling system

The geometric core includes three parts of the CAD system component. There are primitive modeling, operation definition and complex model construction. Primitive modeling part is responsible for creation of new primitives, their storage and modification. The main routines of this part are implemented as the `Primitive` class shown in Figure 4-2. These primitives transform into more complex object, called `Structures` in Figure 4-2, via modeling operations. Several basics operations can be used to create a structure: set-theoretic union, difference, intersection and blended union. User can introduce some specific operations within the primitive definition.

Since FRep exploits defining functions to describe solids, the geometric core needs to operate with symbolic information. The user writes JavaScript code to introduce a new primitive that can be used for visualization and slicing. The system analyzes this code to extract parameters of the primitive and prepare it for further usage.

Esprima Ariya Hidayat [2022] was used to process JavaScript code. It allows us to parse the code and handle its statements in a specific way. Surely, it is necessary to introduce some rules for the code writing if the user wants to obtain a predictable result. Thus, user must specify primitive parameters with the `let` statement. Then, the system extracts them with Esprima and will use the assigned values as default for parameters. Moreover, the special comment construction can be used to specify limits of parameter values. User can add a comment in format of `//SLIDER;[a];[b];[step]` with explicit numerical values instead of `[a]`,`[b]` and `[step]`. Then, the system restricts the possible range of the defined parameter with `[a]` from the bottom and `[b]` from the top. In this case, the parameter can change its value within this range with the specified step `[step]`.

Let us consider an example of a ball primitive definition. One of the possible ball defining functions is:

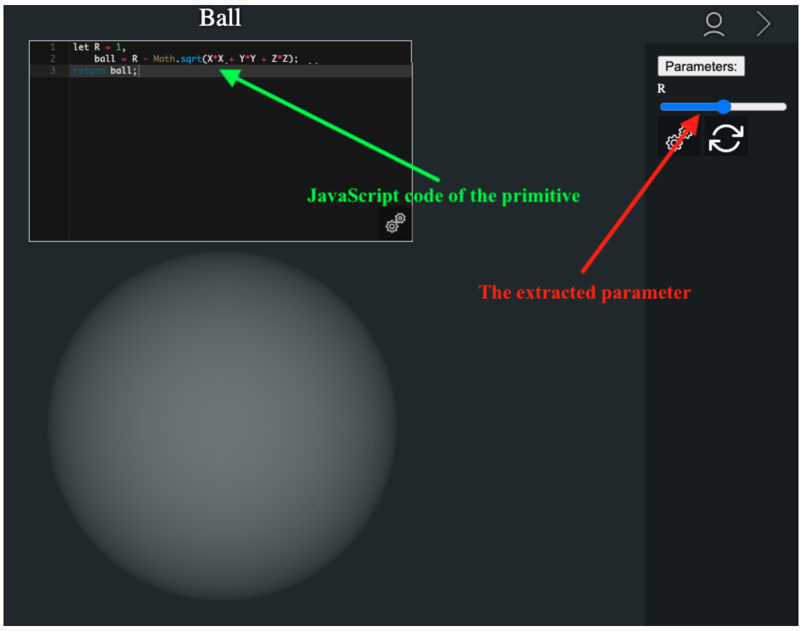
$$f(x, y, z) = R - \sqrt{x^2 + y^2 + z^2},$$

where R is the radius parameter of the ball and the ball is in the origin.

The JavaScript code corresponding this primitive looks as follows:

```
    let R = 1,
        ball = R - Math.sqrt(X*X + Y*Y + Z*Z);
    return ball;
```

The system identifies `R` as a parameter in the above written code. Since, there is no explicit definition of limits, the system applies the default range for the parameter $R \in [0; 2]$.



```
let R = 1,
    ball = R - Math.sqrt(X*X + Y*Y + Z*Z);
return ball;
```

Figure 4-3: Defining of the ball primitive via user interface of the system.

Figure 4-3 shows the above primitive definition in the developed system. The left upper window has a code editor, where user can write the defining function. Its parameters appear in the right control panel after processing. The range slider placed in this panel allows user to change radius parameter R within $[0; 2]$ with the default step. It equals 0.01.

The system has several builtin operations that are used for constructing complex objects. For example, users can exploit blending union:

$$b(f_1, f_2) = f_1 + f_2 + \sqrt{f_1^2 + f_2^2} + \frac{a_0}{1 + \left(\frac{f_1}{a_1}\right)^2 + \left(\frac{f_2}{a_2}\right)^2}.$$

The code of this operation looks like:

```
float Blend(float obj1, float obj2, float a0, float a1, float a2){
    float un = obj1 + obj2 + sqrt(obj1 * obj1 + obj2 * obj2);
    float displacement = a0 / (1. +
        (obj1 / a1) * (obj1 / a1) + (obj2 / a2) * (obj2 / a2));
    return un + displacement;
}
```

where `obj1` and `obj1` are values of defining functions f_1 and f_2, and a_0, a_1, a_2 are control parameters of the blending operation.

You can see how these parameters affect the result solid in Figure 4-4. This figure shows the result of blending union applied to two balls defined by f_1 and f_2 with equal radii and different center points. Figure 4-4(a) presents the result of pure union operation. You can see how extra material appears between the balls when a_0 is assigned with a positive value in Figure 4-4(b). The bigger this value, the more material added to the result object as you can see in Figure 4-4(c). User can control material contribution of each primitive by changing a_1 and a_2 as shown in Figure 4-4(d).

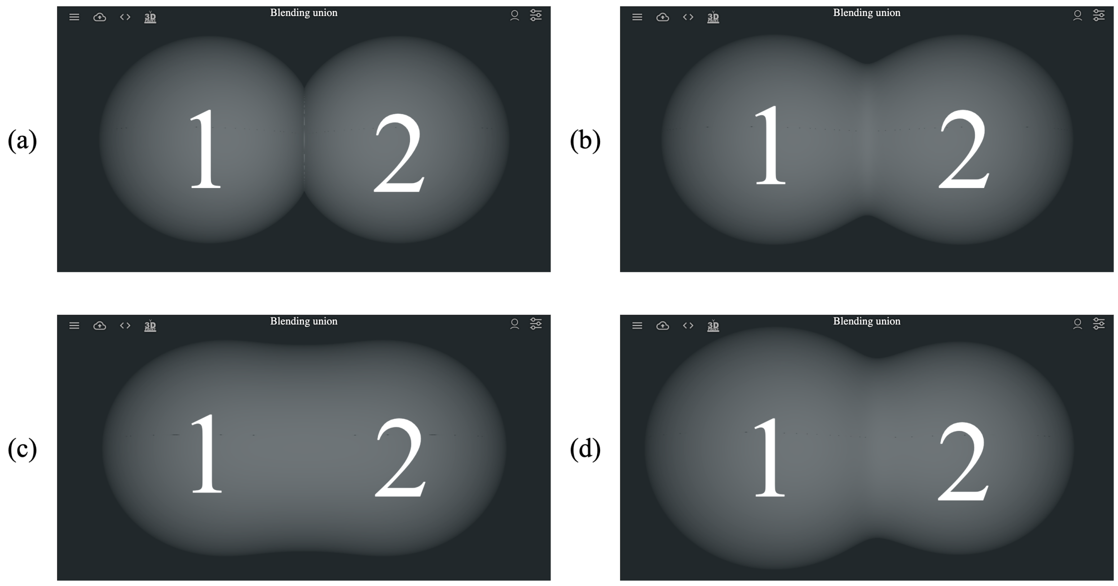


Figure 4-4: The blending union operation with different parameters: (a) $a_0 = 0$, thus it is equivalent to simple union; (b) $a_0 = 0.03, a_1 = 0.1, a_2 = 0.1$; (c) $a_0 = 0.16, a_1 = 0.1, a_2 = 0.1$; (d) $a_0 = 0.03, a_1 = 0.1, a_2 = 0.46$.

The developed system actively uses GPU and, therefore, the main goal of the geometric core is a translation of defining functions written in JavaScript into proper

GLSL codes. One of the constantly used algorithms is rendering which is applied for object visualization.

4.3 Rendering of FRep objects

Figure 4-5 shows the general scheme of the proposed rendering algorithm. It introduces the 3D modeling space $\mathcal{M}$ which includes FRep objects. The following description of the rendering algorithm operates with entities in this space.

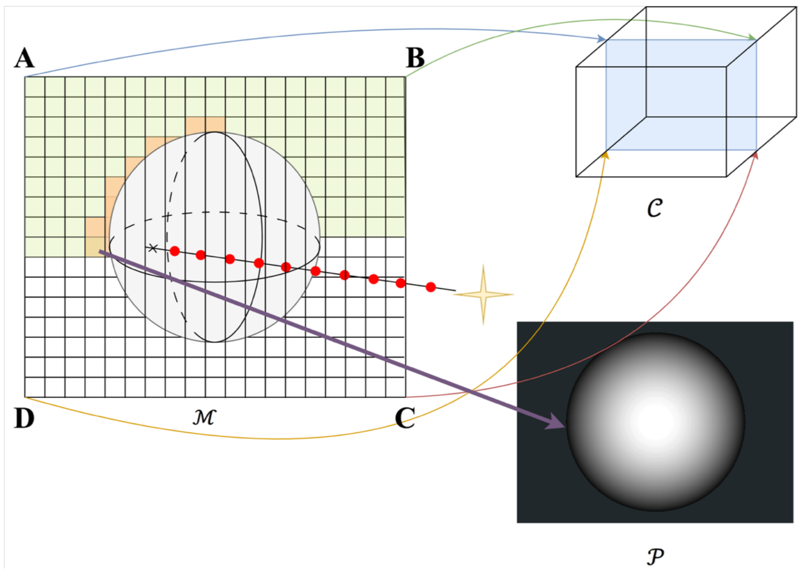


Figure 4-5: Rendering scheme.

The yellow star in the center of Figure 4-5 is a point of view or a camera. It looks at the center of the rectangle $ABCD$. This rectangle limits the projection plane used for the rendering process. It has the same number of attraction points as the number of pixels in the HTML canvas element used for drawing, and it preserves its proportions in vertical and horizontal directions.

The rendering algorithm suppose a marching procedure along a bunch of rays emitted from the point of view until tracing points reach the boundary of the rendered object. The line between the point of view and an attraction point at the projection plane shows one of rays in Figure 4-5.

The step size of this process is quite important. One can use a constant step size for rendering but it leads to the trade-off between accuracy of the rendering and time efficiency. The smaller step size is, the larger number of steps is required to process the same line segment. The constant step size approach is more stable and obvious, however, it is more reasonable to use an adaptive step size from the practical perspective. The algorithm uses the step size depending on the FRep defining function in the proposed system. The chosen step size equals the minus value of the defining function at the current point.

It leads to several consequences. The first one is that for infinite number of steps the algorithm stops at the boundary of object because its defining function is equal to zero there. Since one object can be described via unlimited number of defining functions, one can chose a function that provides better rendering results. For example, one can affect the speed of marching via multiplying the function by a constant. Moreover, the practice shows that functions with properties similar to signed distances allow rendering algorithm to find the boundary faster.

Moreover, one can improve efficiency of rendering for complex objects by tuning the defining functions of used primitives. For example, if the result defining function f includes primitive defining functions $f_1, ..., f_m$, then one can introduce constant multiplies $a_1, ..., a_m$ and use modified primitive functions $a_1 f_1, ..., a_m f_m$ to improve rendering efficiency. These techniques allow us to obtain the boundary detection in convenient number of steps. The implemented rendering algorithm in the proposed system performs 30 steps of the ray marching procedure. It appears enough for an appropriate rendering of currently modeled objects.

When the rendering algorithm finds the intersection point with a boundary, it chooses the color of the corresponding pixel. The algorithm can draw closer boundary points with more bright colors (see Figure 4-6(a)), or can numerically calculate the gradient of the defining function at a boundary point to simulate the reflection of a surface (see Figure 4-6(b)).

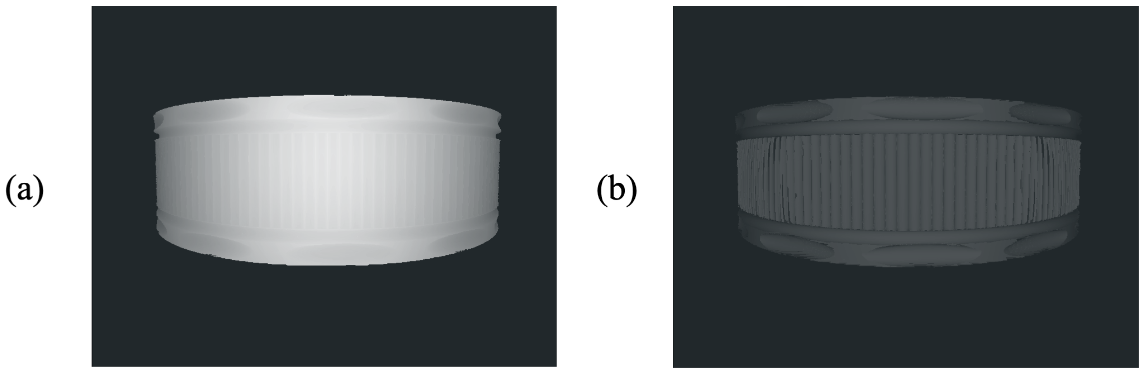


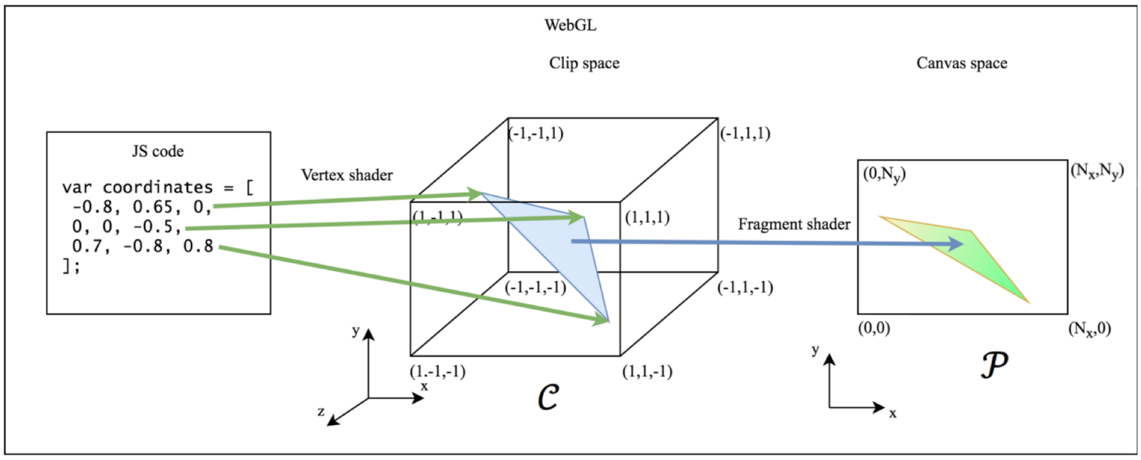
Figure 4-6: Rendering results of a ring model: (a) without reflection; (b) with reflection.

The proposed rendering algorithm can be adopted for the models rendering with variable material properties, like composite material [Prajapati et al., 2022]. The FRep concept assumes that an object description can have additional real-valued functions. These functions define object properties, such as material, textures, and opacity. The rendering algorithm can be updated for using these functions in objects' color calculations.

WebGL The Khronos Group Inc [2022] library is used for running GLSL rendering codes. WebGL is a part of the HTML canvas element since HTML5 (Tim Berners-Lee [2022]) standard issued. It serves to draw 2D and 3D shapes in the canvas element with GPU. This tool assumes that the drawn objects consist of textured portions of surfaces, e.g. triangles, as in the ".STL" file format. There are two GLSL programs exploited by WebGL. The first one is a vertex shader which operates with portions of the boundary. The second one is a fragment shader which performs texturing of these portions.

Figure 4-7 shows how this tool works. It starts from processing JavaScript code which defines the geometrical primitives being processed with a vertex shader. It can process points, lines and triangles. The left part of the Figure 4-7 shows the definition of a triangle as a set of its 3D vertices. The vertex shader works in the coordinate system of the clip space. This space is a $2 \times 2 \times 2$ cube with coordinates changing form -1 to 1. You can see how the vertex shader placed the defined triangle into the clip space $\mathcal{C}$ in the middle of Figure 4-7. Finally, WebGL plots textured geometry in the HTML canvas element. A fragment shader performs texturing of shapes created by the vertex shader as shown in the right part of Figure 4-7.

These two shaders have two spaces associated with them. The vertex shader works in the 3D clip space $\mathcal{C}$ described above. The 2D picture space $\mathcal{P}$ is associated with the fragment shader. The HTML canvas element defines its limits: $x_{\mathcal{P}} \in [0, N_x]$ and $y_{\mathcal{P}} \in [0, N_y]$, where N_x is the width and N_y is the height of the canvas. Moreover, $x_{\mathcal{C}}$ and $y_{\mathcal{C}}$ coordinates of the space $\mathcal{C}$ have the following connection with coordinates $x_{\mathcal{P}}$ and $y_{\mathcal{P}}$ of the picture space $\mathcal{P}$:



```
var coordinates = [
 -0.8, 0.65, 0,
 0, 0, -0.5,
 0.7, -0.8, 0.8
];
```

Figure 4-7: WebGL operation scheme.

$$x_{\mathcal{C}} = \frac{x_{\mathcal{P}} - \frac{N_x}{2}}{\frac{N_x}{2}}, x_{\mathcal{P}} = (x_{\mathcal{C}} + 1)\frac{N_x}{2},$$
$$y_{\mathcal{C}} = \frac{y_{\mathcal{P}} - \frac{N_y}{2}}{\frac{N_y}{2}}, y_{\mathcal{P}} = (y_{\mathcal{C}} + 1)\frac{N_y}{2},$$

e.g. a point with $x_{\mathcal{C}}$ and $y_{\mathcal{C}}$ coordinates equal to $(-1, -1)$ of the clip space C corresponds to the point $(0, 0)$ of the picture space $\mathcal{P}$; and a point with $x_{\mathcal{C}}$ and $y_{\mathcal{C}}$ coordinates equal to $(1, 1)$ of the clip space C corresponds to the point (N_x, N_y) of the picture space $\mathcal{P}$.

Since WebGL is oriented to BRep geometry the algorithm specially adapted it for FRep rendering. The vertex shader performs mapping of the projection plane $ABCD$ onto the plane with coordinates $(-1, 1, 0), (1, 1, 0), (1, -1, 0)$ and $(-1, -1, 0)$ in the clip space $\mathcal{C}$ as shown in Figure 4-5. This plane serves as a canvas for drawing the isometric projection of objects which the fragment shader performs.

One more thing should be encountered during constructing a rendering algorithm is the discrete nature of the picture. The HTML canvas element consists of pixels. Thus, the work of shaders has visible results only in discrete number of space points. For example, the bottom left pixel of the canvas has coordinate $(0.5, 0.5)$ in the picture space $\mathcal{P}$ coordinates by the default.

Chapter 5

Optimization of FRep objects

The biggest advantage of AM is, probably, freedom in shapes that can be manufactured. FRep itself has huge power in the designing of complex objects. However, there is one more source of not only complex but also functional geometry. This is topology optimization (see Section 2.2). FRep can be successfully used with this method.

This chapter delivers the results mainly presented in [Popov et al., 2021a]. Here the thesis discusses the connection of level set method for topology optimization with FRep and propose an efficient topology optimization algorithm for the compliance minimization problem.

5.1 The proposed topology optimization algorithm

The developed system uses the modified topology optimization algorithm presented in [Wei et al., 2018]. The compliance minimization problem used in the system can be expressed as follows:

$$
\begin{aligned}
\min_{f} \quad & \int_D E_{ijkl}\varepsilon_{ij}(\boldsymbol{u})\varepsilon_{kl}(\boldsymbol{u})H(f)d\Omega, \\
\text{s.t.} \quad & a(\boldsymbol{u},\boldsymbol{v},f) = l(\boldsymbol{v},f), \quad \forall \boldsymbol{v} \in U, \\
& \boldsymbol{u}|_{\Gamma_1} = \boldsymbol{u_0}, \\
& E_{ijkl}\varepsilon_{ij}(\boldsymbol{u})|_{\Gamma_2} = \boldsymbol{p}, \\
& \int_D H(f)d\Omega \le V_0,
\end{aligned}
\tag{5.1}
$$

where

$$
\begin{aligned}
a(\boldsymbol{u},\boldsymbol{v},f) &= \int_\Omega E_{ijkl}\varepsilon_{ij}(\boldsymbol{u})\varepsilon_{kl}(\boldsymbol{v})H(f)d\Omega, \\
l(\boldsymbol{v},f) &= \int_\Omega \big(\boldsymbol{g}\boldsymbol{v} + \nabla\left(\boldsymbol{p}\boldsymbol{v}n\right)\big)H(f)d\Omega,
\end{aligned}
$$

D is the design domain including the optimized body Ω; $H(f)$ is the Heaviside function that represents the presence of material at a point of the level-set function f:

$$
H(f) = \begin{cases} 1 & \text{if } f \ge 0, \\ 0 & \text{if } f < 0. \end{cases}
$$

This problem statement uses the level-set function f which, at the same time, is an FRep defining function (Equation 2.1).

Let us consider the 2D optimization case. The proposed topology optimization algorithm incorporates a free-form defining function with bilinear basis functions:

$$
\phi(x,y) = \begin{cases} (1-|x|)(1-|y|) & \text{if } |x| \le 1 \text{ and } |y| \le 1, \\ 0 & \text{otherwise.} \end{cases}
\tag{5.2}
$$

This function defines a surface drawn in Figure 5-1(a). Moreover, in general case, one need to use this equation with constants h_1 and h_2 to define a rectangular non-zero region of the spline basis function, as follows:

$$\phi'(x,y)=\begin{cases}(1-\frac{1}{h_1}|x|)(1-\frac{1}{h_2}|y|) & \text{if } |x|\le h_1 \text{ and } |y|\le h_2,\\ 0 & \text{otherwise.}\end{cases}$$

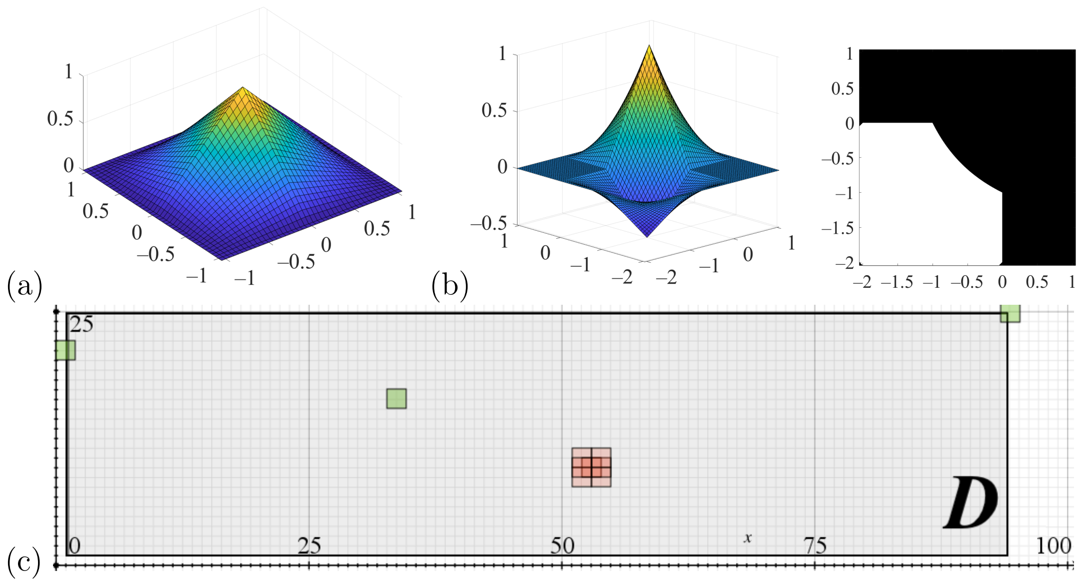


Figure 5-1: Bilinear spline construction: (a) the surface defined by the bilinear spline basis function (Equation 5.2); (b) the surface produced by two spline basis functions and its zero-level set (boundary); (c) rectangular domain D with 90 $\times$ 25 elements and 91 $\times$ 26 knots.

However, the form (Equation 5.2) can be used here without the loss of generality. This form of the spline basis function is well suited for the further discussion. Space mapping is used to locate spline basis function at some portion of the domain. The bilinear spline basis function with the center of the non-zero region at (x_i, y_i) is:

$$\phi_i(x,y)=\begin{cases}(1-|x-x_i|)(1-|y-y_i|) & \text{if } |x-x_i|\le 1\\ & \text{and } |y-y_i|\le 1\\ 0 & \text{otherwise.}\end{cases}$$

Therefore, the free-form defining function used here can be expressed as follows:

$$f(x,y)=\sum_{i=1}^{N}\alpha_i\phi_i(x,y) \tag{5.3}$$

with constants α_i.

The main steps of the proposed modified algorithm are shown in Algorithm 1.

Algorithm 1: The proposed topology optimization algorithm

```
Step 1. Define the number of grid elements in the domain $D$;
Step 2. Initialize coefficients of the free-form function;
Step 3. Define boundary conditions for FEM analysis;
Step 4. Initialize parameters of the optimization loop;
Step 5. Perform FEM analysis of the domain;
Step 6. If the algorithm converges, then quit, else update coefficients of the
  free-form function and go to Step 5.
```

The algorithm uses Equation 2.2 to update the free-form function defined by Equation 5.3. The equation for updating α_i is as follows:

$$\alpha_i(t_{j+1}) = \overline{\alpha}_i(t_j) + \Delta t \Phi^{-1} B(\overline{\alpha}_i(t_j), t_j),$$

where

$$\overline{\alpha}_i(t_j) = \frac{\alpha_i(t_j)}{mean(|f(x_1, y_1, t_j)|, |f(x_2, y_2, t_j)|, ..., |f(x_r, y_r, t_j)|},$$
$$\Phi = \begin{pmatrix} \phi_1(x_1, y_1) & \cdots & \phi_N(x_1, y_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_N, y_N) & \cdots & \phi_N(x_N, y_N) \end{pmatrix}, \tag{5.4}$$

$(x_1, y_1), ..., (x_r, y_r)$ are the points near the solid boundary, $(x_1, y_1), ..., (x_N, y_N)$ are the knot points of the free-form function (Equation 5.3), $B(\overline{\alpha}_i(t_j), t_j)$ is the velocity vector at the time step t_j:

$$B(\overline{\alpha}_i(t_j), t_j) = \begin{pmatrix} V(x_1, y_1, t_j)\delta(f(x_1, y_1, \overline{\alpha}_i(t_j))) \\ \vdots \\ V(x_N, y_N, t_j)\delta(f(x_N, y_N, \overline{\alpha}_i(t_j))) \end{pmatrix}. \tag{5.5}$$

For the compliance minimization problem (Equation 5.1), the normal velocity V can be derived as follows:

$$V = E_{ijkl}\varepsilon_{ij}(\boldsymbol{u})\varepsilon_{kl}(\boldsymbol{u}) - \lambda$$

where λ is the Lagrange multiplier to deal with the constraint of the volume fraction, which is calculated using the following augmented Lagrangian updating scheme:

$$\lambda^{k+1} = \begin{cases} \mu G^k & k \leq n_R \\ \lambda^k + \gamma^k G & k > n_R \end{cases}, \tag{5.6}$$

where μ and γ^k are parameters in the k-th iteration of the optimization, and γ^k is updated using the following scheme:

$$\gamma^{k+1} = min(\gamma^k + \Delta\gamma, \gamma_{max}) \quad k > n_R.$$

where $\Delta\gamma$ is the increment and γ_{max} is the upper limit of the parameter γ. Since the volume fraction of initial design usually does not meet the prescribed volume fraction, the volume constraint is relaxed in the first n_R iterations as below:

$$G^k = \int_D H(f)d\Omega - \left(V_0 - (V_0 - V_{max})\frac{k}{n_R}\right) \quad k \leq n_R. \tag{5.7}$$

To avoid the unbounded growth of the free-form function, an approximate $\delta(f)$ function is inserted into the velocity vector of (Equation 5.5), defined as follows:

$$\delta(f) = \begin{cases} 0 & f < -\Delta \\ \frac{3}{4}(1 - \frac{f^2}{\Delta^2}) & -\Delta \leq f \leq \Delta \\ 0 & f > \Delta \end{cases},$$

where the parameter Δ is used to control the magnitude of the upper and lower bounds of the free-form function. The ersatz material model (see Section 2.3) is used in the finite element analysis to obtain the naturally extended velocity field from the strain energy density in the whole design domain. The MATLAB implementation of the proposed optimization algorithm is given in [Popov et al., 2021a].

5.2 Modifications of the algorithm

The free-form function (Equation 5.3) uses bilinear basis functions instead of multiquadric basis proposed in the original algorithm [Wei et al., 2018]. The use of bilinear basis functions has several advantages. First, the α_i from Equation 5.3 has a simple interpretation. The only spline basis function placed at some point (x_i, y_i) with positive coefficient α_i means that its non-zero region $[x_i-1, x_i+1], [y_i-1, y_i+1]$ is filled with material. Otherwise, if α_i is negative, no material is present in $[x_i - 1, x_i + 1], [y_i - 1, y_i + 1]$. Two spline basis functions $\phi_i(x, y)$ and $\phi_j(x, y)$ with shared non-zero regions and corresponding α_i and α_j of opposite signs define a portion of the body in the shared region with the boundary defined by the equation:

$$\alpha_i(1 - |x - x_i|)(1 - |y - y_i|) = -\alpha_j(1 - |x - x_j|)(1 - |y - y_j|). \tag{5.8}$$

Equation 5.8 defines a conic section in 2D. Figure 5-1(b) shows the case of two bilinear spline basis functions with the shared region and corresponding α_i and α_j of opposite signs. The black region on the right of this picture is the domain where the considered function is positive.

Second, the algorithm proposed in [Wei et al., 2018] works with a rectangular domain D and the bilinear spline (Equation 5.3) can accurately describe it. Consider the domain as shown in Figure 5-1(c) covered by 91×26 spline basis functions. Three non-zero regions of the bilinear spline basis functions are shown in green, seven shared non-zero regions of the bilinear spline basis functions are shown in red. Grid knots are the centers of their non-zero regions. Every non-zero region of a spline basis function covers four white regions. However, it is not the case for the basis functions placed at the boundary of D. One can define the whole rectangular domain D by a finite number of spline functions in the form proposed in Equation 5.3.

Two conditions hold for all these functions. The α_i values corresponding to all spline basis functions placed at the boundary, equal zero, and others are positive. The optimization algorithm requires the free-form function (Equation 5.3) to have properties of a signed distance function near the boundary. Therefore, for example, for the basis functions placed at the boundary, α_i can be initialized with zeros, and in other cases — with ones.

One more reason to use bilinear splines in Equation 2.1 is the FEM step. At this step, the Young's modulus of the rectangular element is approximated proportionally to the element's density. For the elements with no material, the Young's modulus is assumed to be zero. Young's modulus of elements filled with material is assumed to be equal to that of proposed material. The algorithm estimates the density of elements partially located at the boundary. It uses exact values of the bilinear spline (Equation 5.3).

Another modification of the algorithm from [Wei et al., 2018] relates to Steps 2 and 6 of the initial optimization algorithm. In the examples from [Wei et al., 2018], the defining function initially has positive values at the boundary of the domain D. From the FRep perspective, it means that the body has a surface somewhere outside D. However, it is not correct. Mathematically, α_i at the boundary of the domain D has to be zero or negative.

Suppose at some iteration `i` of the algorithm one obtained the defining function $f_i(x, y)$. The idea above can be written as:

$$f_i(x_j, y_j) = min(f_{i-1}(x_j, y_j), 0) \tag{5.9}$$

where (x_j, y_j) is a boundary point of D.

The rule introduced above has an exception. Consider the boundary conditions for the case shown in Figure 5-2(a), left. The shown body has three special points. There are two points where the body touches rollers and the point where force F is applied. Let us suppose that another body produces this force. Therefore, near these three points, there is a region where optimized touches external bodies. These points always contain a portion of material. Thus, Equation 5.9 does not hold for them.

One implementation issue should be noted (see the MATLAB code in Popov [2022b]). Consider the initialization of the bilinear spline defining the body shown in Figure 5-2(a), right. It is a meaningful option of the body design, but it has holes at the boundary of D. Small negative values are used in Equation (Equation 5.9)

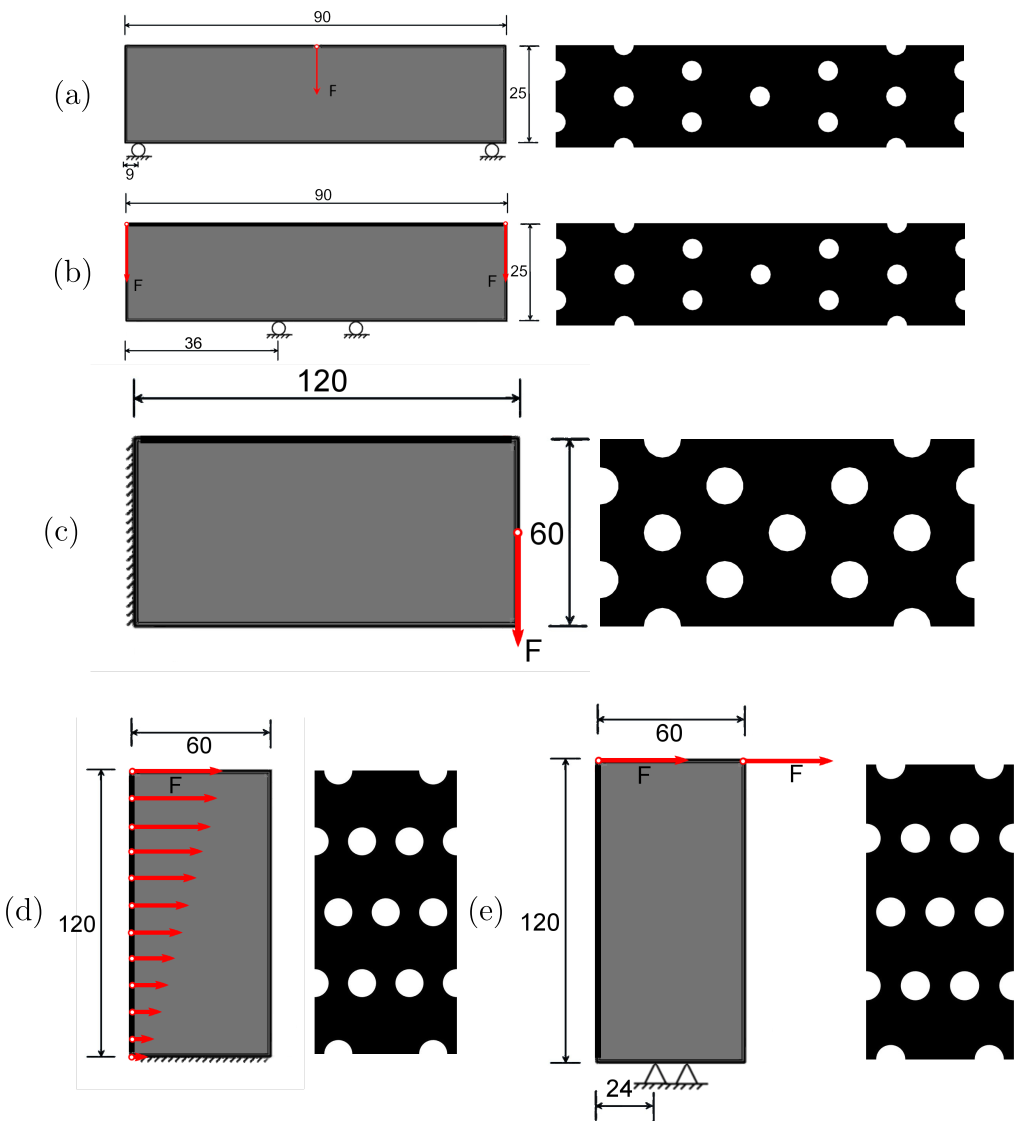


Figure 5-2: Optimization cases and initial assumptions: (a) the three-point bending beam case; (b) the beam is supported horizontally in two bottom points, and equal loads applied vertically in upper corners; (c) the cantilever beam case; (d) the beam is fixed along the bottom side, and the horizontal load linearly distributed along the left side of the beam; (e) the beam is fixed horizontally in two bottom points, and equal loads applied horizontally in upper corners.

instead of zero to allow the algorithm to go over similar shapes.

This modification of Equation 5.9 is also useful in one more situation. During the optimization process, at some iteration i, one can obtain the defining function (Equation 5.3) with all positive α_j. It is true for all points except ones mentioned in (Equation 5.9). Replacement of zero with some small negative value prevents the algorithm error at the normalization step (Equation 5.4). The above-mentioned modifications make the algorithm more robust.

The proposed modified topology optimization algorithm was compared with the original one [Wei et al., 2018]. The laptop with the following characteristics was used for tests: Windows 10 Enterprise 64-bit operating system, x64-based processor Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz 1.80 GHz, and 8.00 GB (7.86 GB usable) of installed memory (RAM). Both algorithms, the original one and the modified one, were run 10 times with following initial data sets. The domain grid is as shown in Figure 5-1(c). The initial geometry is as shown in Figure 5-2(a), right. Boundary conditions are taken from Figure 5-2(a), left (see details in Popov [2022b]). The desired volume, V_0, was chosen as 45% of the volume of the entire D. The Young's modulus of stainless steel, E_0, is assumed to be 205 GPa, and the Poisson's ratio, ν, is assumed to be 0.27. Stainless steel is a material obtained of the powder used for 3D printing. The properties of material were studied in [Kuzminova et al., 2019]. In the original algorithm all parameters of the updating scheme for $f_i(x, y)$ were preserved, except the time step. The time step was changed to 0.2. These parameters were also used for the modified algorithm.

Both algorithms exploit an isotropic material model at the FEM step. However, the research [Kuzminova et al., 2019] showed that 3D printed parts have anisotropy. Nevertheless, the validation results (see Section 5.3) showed that the isotropic model of material is suitable for the considered cases.

Results of the computational test are shown in Table 5.1. The modified algorithm is approximately twice as fast as the original one for the same accuracy concerning the computational grid resolution. It can be easily explained using different spline basis functions. When the bilinear spine is used instead of the multiquadric-spline, the time-consuming operation of matrix inversion is not required. Thus, the modified algorithm works faster for the same number of knot points in the basis function. The reason is that it has linear complexity competing with the quadratic complexity of the original algorithm. Two obtained optimized bodies are shown in Figure 5-3(a).

Four more optimization cases shown in Figure 5-2 were considered to check the observed tendency. More interesting models with complicated lattice structures or cellular structures can be found in [Nazir et al., 2022]. Figure 5-3 shows the results

Test Number	Modified Algorithm, sec	Wei's Algorithm, sec
1	36.8	80.8
2	35.2	77.3
3	36.1	81.7
4	35.9	75.8
5	35.2	76.6
6	35.9	77.3
7	35.4	76.2
8	35.5	76.4
9	35.9	76.1
10	36.3	76.8
Average time, sec	**35.8**	**77.5**
Number of iterations	**213**	**179**
Objective function	**1.71×10^5**	**1.71×10^5**
Volume	**0.45**	**0.45**

Table 5.1: Results of algorithms comparison.

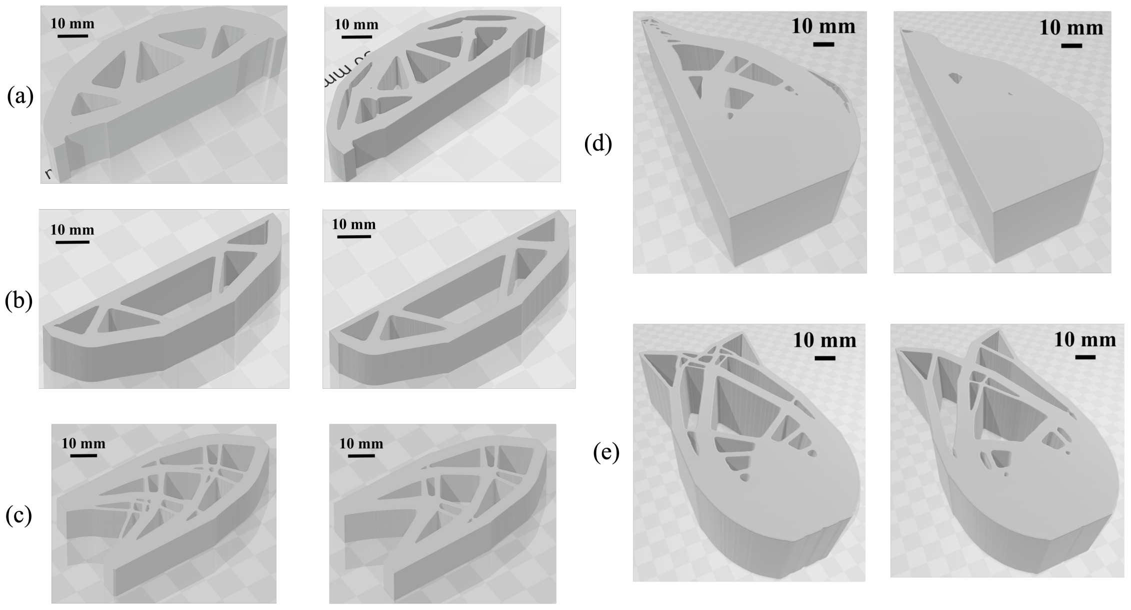


Figure 5-3: Optimal geometries obtained with modified algorithm (left) and original algorithm (right: (a) the three-point bending beam optimization case; (b) the optimization case where the beam is supported horizontally in two bottom points, and equal loads applied vertically in upper corners; (c) the cantilever beam optimization case; (d) the optimization case where the beam is fixed along the bottom side, and the horizontal load linearly distributed along the left side of the beam; (e) the optimization case where the beam is fixed horizontally in two bottom points, and equal loads applied horizontally in upper corners.

of optimization algorithms testing. The efficiency of the algorithms for these cases is compared in Table 5.2. The results show that the modified algorithm is faster.

Case	Time		Objective Function	
	Modified Algorithm, sec	Wei's Algorithm, sec	Modified Algorithm	Wei's Algorithm
Figure 5-2(b)	21.9	51.9	1.06×10^6	1.06×10^6
Figure 5-2(c)	22.2	448.6	6.00×10^5	6.02×10^5
Figure 5-2(d)	165.2	14902.2	1.47×10^{10}	1.59×10^{10}
Figure 5-2(e)	24.4	437.1	2.48×10^7	2.48×10^7

Table 5.2: Efficiency of algorithms for four additional cases.

The difference in efficiency is much higher for finer grids of basis functions in the domain. The original algorithm became 20 times slower than the modified algorithm at the case shown in Figure 5-3(c), which can be explained by the matrix inversion used in the Wei's optimization.

At the same time, both algorithms produce similar values of the objective function, differing in the third significant digit, depending on the optimization case. It does not sound obvious recalling different basis functions used in algorithms. Multiquadric splines can propose better quality of boundary approximation than bilinear splines. However, analysis from [Bendsoe and Sigmund, 2003] shows that better solutions to the considering optimization problem are connected with the complexity of topology rather than with shape accuracy. Therefore, since both basis functions are radial basis functions, topology depends on the number of knot points. Thus, the number of knot points affects the objective function more than the local quality of the shape.

Small modifications of the algorithm from Popov [2022b] were used to produce models shown in Figure 5-3. These modifications are also listed at Popov [2022b].

More accurate optimization takes more time and requires more RAM installed. Both tasks, slicing and optimization, lead to the solution of the trade-off between precision and time. Although calculations on the mentioned laptop provide valid data.

5.3 Algorithm validation

The part shown in Figure 5-3(a) left was printed, and its mechanical tests were conducted to validate the proposed algorithm. Dimensions of the printed part (see Figure 5-4(a)) are $86 \times 25 \times 20$ mm. The part was cut from the substrate and tested in the three-point bending mode (see Figure 5-4(b)) at the Instron 5969 testing machine.

Results of the experimental validation are presented as load-displacement curves for several control points. The vertical component of displacement is used in these curves. The obtained load-displacement curve for point P3 (see Figure 5-4(c)) is presented as red dots in Figure 5-4(f). Computed results for the same load are presented as the solid red line. One can clearly see the disagreement between computational and experimental data.

This discrepancy between the analytical and experimental results of the red lines can be attributed to flutes (see Figure 5-4(c)) manufactured to increase the stability of the part during the test. A sagging was observed in their areas (see Figure 5-4(d)) during the testing process. It influenced the load-displacement curve with the displacement value of the whole printed part. The vertical displacement of point P3 was calculated relative to point P0 (see Figure 5-4(c)) for subtracting the influence of sagging from the load-displacement test curve. Figure 5-4(f) shows the load-displacement curves relative to point P0 as green patterns. The correlation between the experimental and predicted results holds up to a load of 15 kN.

During the compression test, it was possible to achieve local plastic deformations. Depending on the distance from P0, the contribution of plastic deformation to the vertical displacement differs. Therefore, additional points, P1 and P2 (see Figure 5-4(c)), were considered. The computational results for those points are presented in Figure 5-4(e). The correlation between the experimental and computational results for the point P2 is preserved up to the load of 10 kN. For the point P1 the correlation with computational results can be observed up to the load of 5 kN.

Thus, the contribution of plastic deformation is the highest in the vicinity of the point P0. The deflection from the elasticity is observed for point P1. It influ-

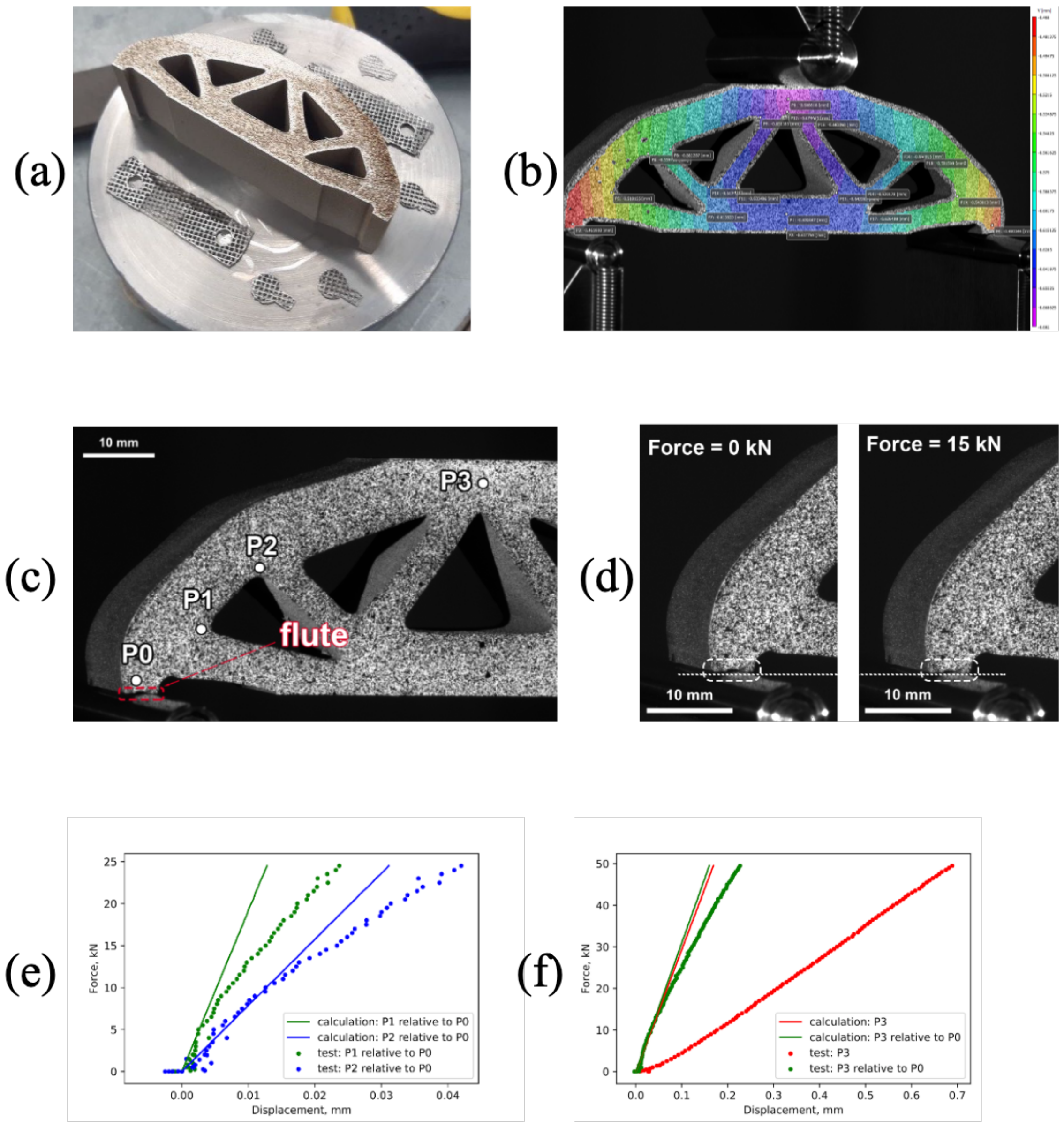


Figure 5-4: (a) The manufactured part attached to the cylindrical substrate. (b) The image from Digital Image Correlation System. (c) The location of tracked points on the part. (d) The test photos of the part under loads 0 and 15 kN. (f) The test and calculation results of vertical displacement for points P1 and P2. (e) The test and calculation results of vertical displacement for point P3.

ences the vertical displacements of points P2 and P3. Therefore, mechanical testing results repeat the theoretical results within the elastic field region, where the computational model was applied. At the loads exceeding 15 kN, the contribution of plasticity starts to grow, and it becomes impossible to compare the computational and experimental results obtained during optimization. At the same time, this experimental validation showed that the proposed approach can be freely used for 3D printed parts that are made up of ceramics [Safonov et al., 2020] and other materials without plastic deformation.

Moreover, the experimental value of the work done by the applied force can be compared and the theoretical values of elastic deformation energy. The upper limit of elastic deformations was observed at the point P3 at the load of 15 kN. This point lies in the vicinity of the loaded point of the body. Therefore, an approximate value of the work done by the applied can be calculated force as follows:

$$A = \frac{Fd}{2} \tag{5.10}$$

where F is the load and d is the deformation at a given point. The load equals 15 kN, d equals 0.053 mm. Thus, the work A equals 0.4 J. The FEM routine of Algorithm 1 provides the same value of elastic energy at these load and dimensions of the printed part.

5.4 Topology optimization in 3D case

Let us consider the 3D case topology optimization. Thus, the FRep defining function is $f(x, y, z)$. It is a continuous function with the following properties:

$$f(x,y,z) \begin{cases} > 0 & (x,y,z) \in \Omega, \\ = 0 & (x,y,z) \in \partial\Omega, \\ < 0 & (x,y,z) \notin \overline{\Omega}, \end{cases}$$

where Ω is an open set of points belonging to a solid body, $\Omega \subset D \subset \mathbb{R}^3$.

Therefore, the free-form functions for 3D case has the following form:

$$f(x,y,z) = \sum_{i=1}^{m} c_i \phi_i(x,y,z), \tag{5.11}$$

where c_i are constants and $\phi_i(x, y, z)$ are basis functions. Equation 5.11 is a trilinear spline [Wei et al., 2021] here.

The basis function of the trilinear spline Equation 5.11 is:

$$\phi(x,y,z) = \begin{cases} (1-|x|)(1-|y|)(1-|z|) & \text{if } |x| \leq 1,\ |y| \leq 1 \text{ and } |z| \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

The plot of this function is shown in Figure 5-5. The basis function of the trilinear spline is mirror symmetric. One of its planes of symmetry is Oxz.

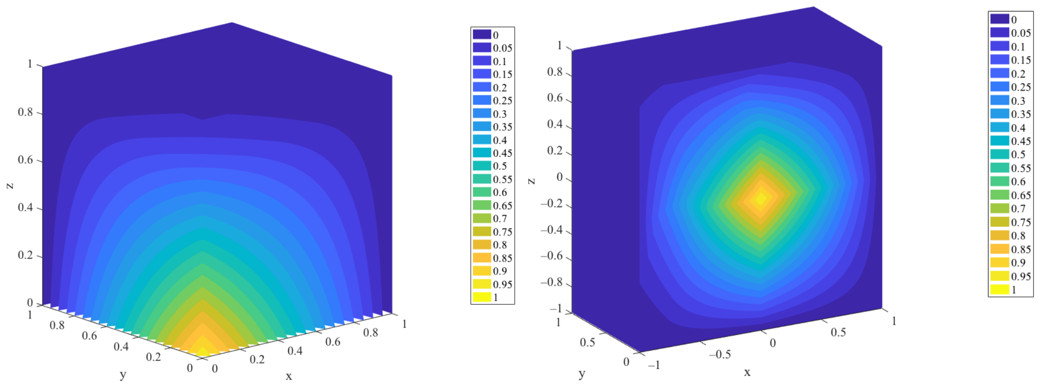


Figure 5-5: Level sets of the bilinear spline basis function.

Functions $\phi_i(x,y,z)$ from Equation 5.11 have the following form:

$$\phi_i(x,y,z) = \begin{cases} (1-\frac{1}{h_x}|x-x_i|)\cdot & \text{if } |x-x_i| \leq h_x, \\ (1-\frac{1}{h_y}|y-y_i|)\cdot & |y-y_i| \leq h_y \\ (1-\frac{1}{h_z}|z-z_i|) & \text{and } |z-z_i| \leq h_z, \\ 0 & \text{otherwise.} \end{cases} \tag{5.12}$$

Therefore, the constants c_i from the equation Equation 5.11 have a simple meaning. Separately positive value of c_i indicates that the box occupying $(x_i - h_x; x_i + h_x)$, $(y_i - h_y; y_i + h_y)$ and $(z_i - h_z; z_i + h_z)$ along the axes x, y and z, respectively, is full of material. The negative value of c_i means the absence of material in the same box. Two overlapping non-zero regions of functions $\phi_i(x,y,z)$ and $\phi_j(x,y,z)$ with coefficients c_i and c_j having different signs define the body surface in this region. Moreover, this surface is a cubic surface.

Consider the free-form function (Equation 5.11) with knot points arranged as shown in Figure 5-6. Figure 5-6(a) shows the rectangular domain D covered with

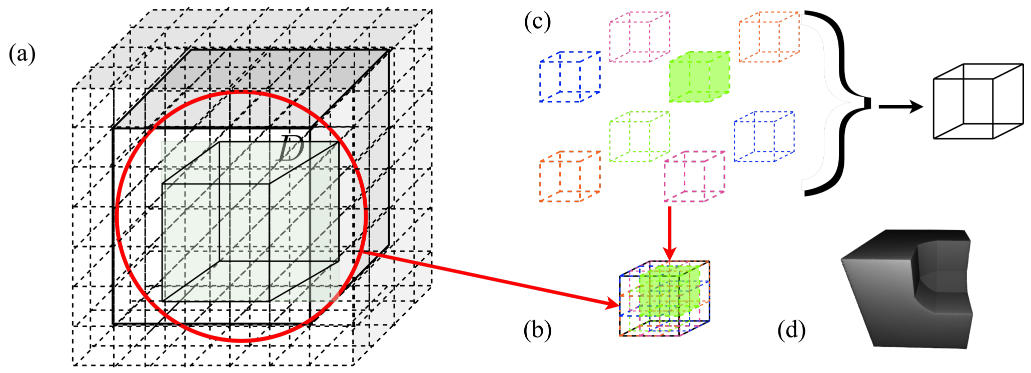


Figure 5-6: Construction of the free-form body: (a) rectangular domain D covered with non-zero regions of spline basis functions; (b) cubic portion of the domain D with eight overlapping non-zero regions of spline basis functions; (c) arrangement of the eight overlapping non-zero regions of spline basis functions; (d) an example of free-form body with seven positive and one negative coefficients.

non-zero regions of basis functions (Equation 5.12). They are drawn with dashed lines. A cubic portion of the domain is shown in Figure Figure 5-6(b). It contains eight overlapping non-zero regions of the basis functions. Their arrangement is drawn in Figure 5-6(c). All of them overlap at the center of the cubic portion. This pattern repeats throughout the domain D.

Figure 5-6(d) shows an example of a free-form body defined by the spline function with only eight basis functions (as shown in Figure 5-6(b)). Seven coefficients in this body are equal $c_1 = c_2 = \cdots = c_7$. The eighth coefficient is negative and has the same absolute value $c_8 = -c_1$. The non-zero region of the basis function with negative coefficient is colored green in Figure 5-6 (b) and (c).

The optimization algorithm (see Algorithm 1) for 3D case works as shown for the example body in Figure 5-7. The MATLAB realization of this optimization algorithm is published in the GitHub repository Popov [2022a]. It has two files and three folders. The optimization algorithm is written in `3D_FRep_optimizer.mlx` live editor script. The file `elemM.mat` contains element stiffness matrix for the 3D finite element method implementation. The MATLAB live editor script `Cubic_-Elastic_element.mlx` is placed in `Cubic_element_matrix` folder. This script was used to calculate the element stiffness matrix saved in `elemM.mat`.

The second folder `Drawing_scripts` contains the script for drawing the geometry

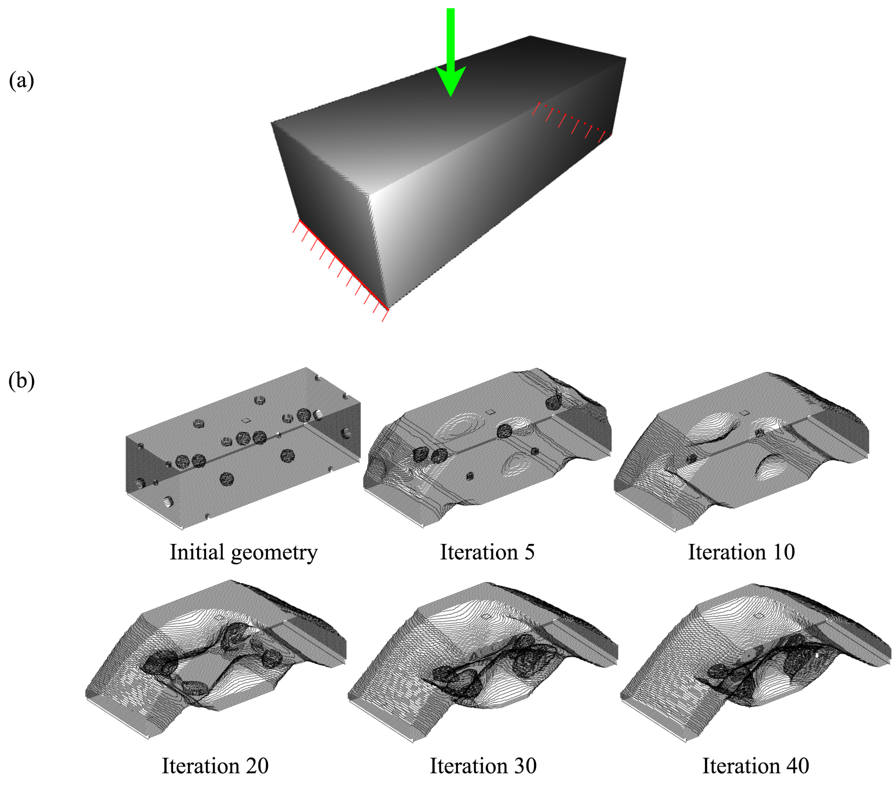


Figure 5-7: Geometry evolution by level set topology optimization method: (a) boundary conditions of the optimization task. where the load is marked with green arrow and fixed portions of the boundary are shown in red; (b) obtained geometries.

in FRep. It generates contours of a body layer by layer and produces the images as shown in Figure 5-7(b).

The `3D_FRep_optimizer.mlx` live editor script has the following structure. The first line defines dimensions of the domain D and the volume constraint V_0. The next section consists of the initialization of the geometry. Then the script has a block to prepare for FEM step. It sets up loaded and fixed nodes and assigns the value of Equation 5.11 at boundary points except these nodes with small negative value (-10^9). This value is the upper limit of the defining function at these points to preserve $\Omega \subset D$. Moreover, the algorithm creates variables to store FEM data at this step. It defines elastic properties of a material, i.e. Young's modulus ($E_0 = 1$) and Poisson ratio ($\nu = 0.3$). The initialization ends with defining boundary conditions.

The last part of the script is the optimization loop. The maximum number of iterations, relaxation number n_R, parameters μ and γ are defined for the Lagrangian multiplier updating scheme (Equation 5.6), time step Δt for the coefficient updating scheme and parameter Δ of the approximate function $\delta(f)$ (Equation 5.7). Then the iterative optimization algorithm starts. Each iteration begins with FEM step. The algorithm uses the ersatz material model, where the element Young's modulus is defined as:

$$E = E_{min} + (E_0 - E_{min})\rho_e, \tag{5.13}$$

where E_{min} is a small number ($E_{min} = 10^{-9}$) used to preserve numerical stability outside the optimized body ($(x, y, z) \notin \overline{\Omega}$), ρ_e is an element density calculated using values of f at 11 points of the element.

The optimal criterion of the algorithm consists of three conditions. The first one is $k > n_R$, where k is the number of the iteration, and n_R is the relaxation number. The second one is $\frac{V_k}{V_0} = 1 \pm 10^{-3}$, where V_k is a volume of the optimized body at the k-th iteration. The last condition is $\frac{J_0^j(\boldsymbol{u})}{J_0^k(\boldsymbol{u})} = 1 \pm 10^{-3}$, where $J_0^k(\boldsymbol{u})$ is a compliance energy of deformation in the k-th iteration, and $J_0^j(\boldsymbol{u})$ is a compliance energy of deformation in the iteration $j = k-9, \ldots, k-1$.

The last step of the optimization loop is the free-form function update according to Equation 5.7. Additionally, it controls values of the function at the boundary points in this step. As during the initialization, the algorithm assigns them with a small negative value, e.g. 10^{-9}.

The published version of the algorithm performs the optimization shown in Figure 5-7. It can be modified for different applications. Two examples of such modifications are placed in the `Examples` folder in Popov [2022a].

One of possible applications for the proposed algorithm is the topology optimization of the oil platform support [Tian et al., 2019] (see Figure 5-8). It is a massive structure that holds equipment for the extraction of petroleum. On the one hand, it shall be durable enough. On the other hand, it is cheaper to use less material for its construction.

This task is an optimization problem in the rectangular domain D with a volume constraint equal to 35% of the total volume of the domain. The domain D has the following dimensions: $20 \times 20 \times 64$. The pressure of the petroleum equipment corresponds to the uniform vertical load. It is applied to the top nodes. The

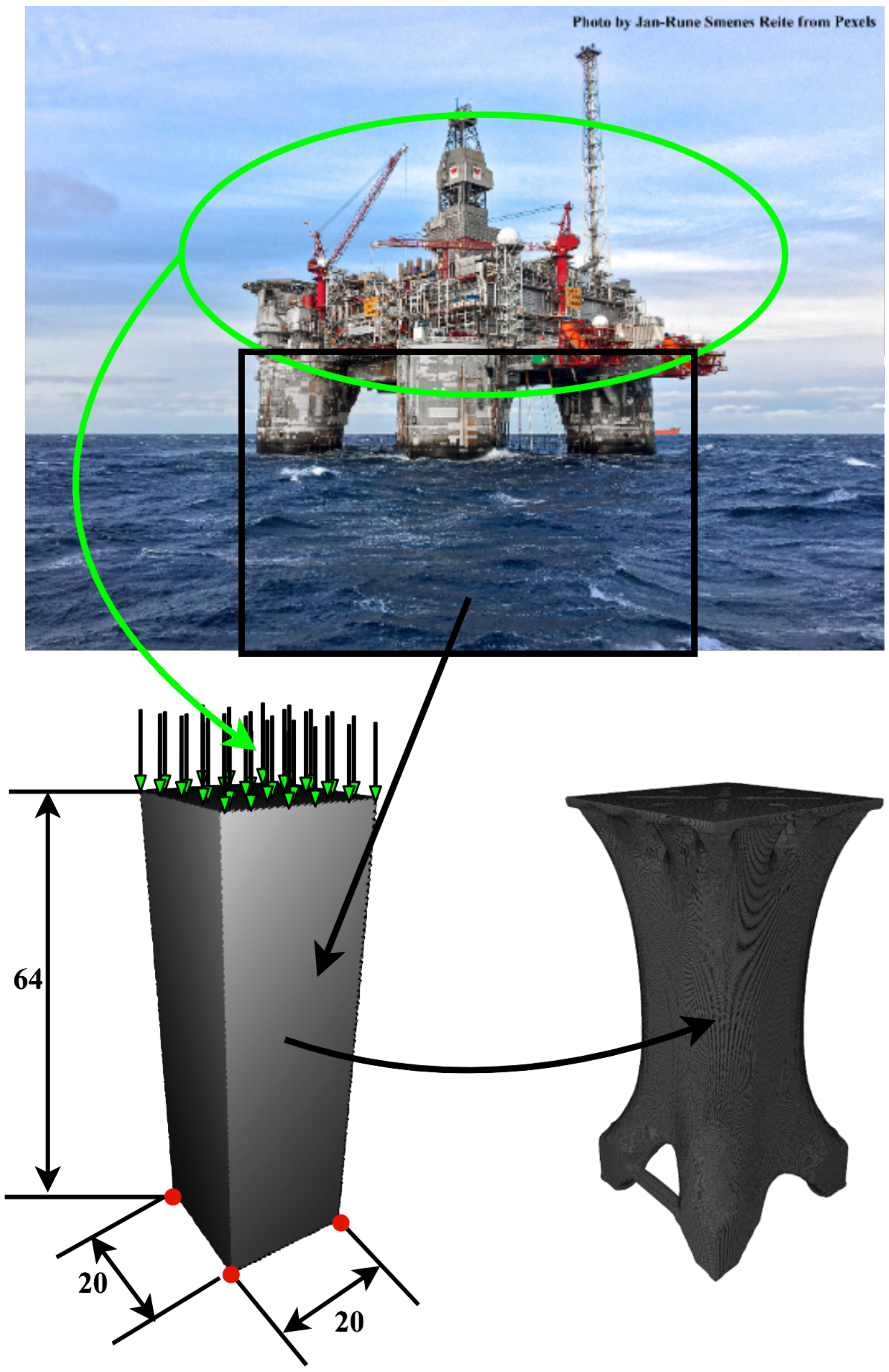


Figure 5-8: Topology optimization of the oil platform support.

construction is fixed at four bottom corner nodes along all directions.

The modified optimization script for the oil platform optimization case is placed into `Examples\3D_platform_optimizer.mlx`

The proposed optimization algorithm can be combined with FRep-based modeling for more complex cases of optimization.

5.5 Shape constraints

Previously mentioned optimization cases supposed that optimization algorithm can freely modify the whole domain. This chapter considers cases when this assumption is wrong. It means that the optimized body has some regions that cannot be modified or they have some restrictions on their modification. Let us start from the example with forbidden optimization of some domain portions.

One need to modify Algorithm 1 for taking into account some constraints on shape changes. Two extra actions appear as follows:

Algorithm 2: The topology optimization algorithm with shape constraints

```
Step 1. Define the number of grid elements in the domain $D$;
Step 2. Initialize coefficients of the free-form function;
**Step 3. Initialize and apply shape constraints to the FRep
  defining function;**
Step 4. Define boundary conditions for FEM analysis;
Step 5. Initialize parameters of the optimization loop;
Step 6. Perform FEM analysis of the domain;
Step 7. If the algorithm converges, then quit, else update coefficients of the
  free-form function, **apply shape constraints** and go to Step 6.
```

These modifications formally affect the FRep defining functions used in the optimization algorithm. It uses complex FRep defining function $f = f(x, y, f_o)$ or $f = f(x, y, z, f_o)$ instead of the free-form function f_o defined by Equation 5.3 or Equation 5.11. The free-form function is incorporated into the result defining function using FRep operations (see Subsection 2.1.2). Thus, the applying of shape constraints at `Step 7` of Algorithm 2 means that the function f does not change except updating of f_o coefficients.

This feature of the proposed optimization framework is shown on the example of a femoral implant. While stiff implants can lead to bone loss, complex revision surgery, and strain shielding, the topology optimization technique aims to develop more compliant implant designs [Tan and van Arkel, 2021], [Abate et al., 2021]. It has been proven that optimized implants bring the stress as close as in an intact femur, especially along the length of implants [Shuib et al., 2005]. According to [Al-Tamimi et al., 2020], the combined use of optimization techniques and additive technologies can help manufacture lightweight personalized orthopedic implants with minimal stress shielding. However, this type of designed product need to fit some features of the human body in contrast with the previously considered examples.

The topology optimization of the femoral implant steam is a representative example of the complex optimization task (see Figure 5-9). The goal was to lighten the most massive portion of the implant that is marked with green rectangles in Figure 5-9.

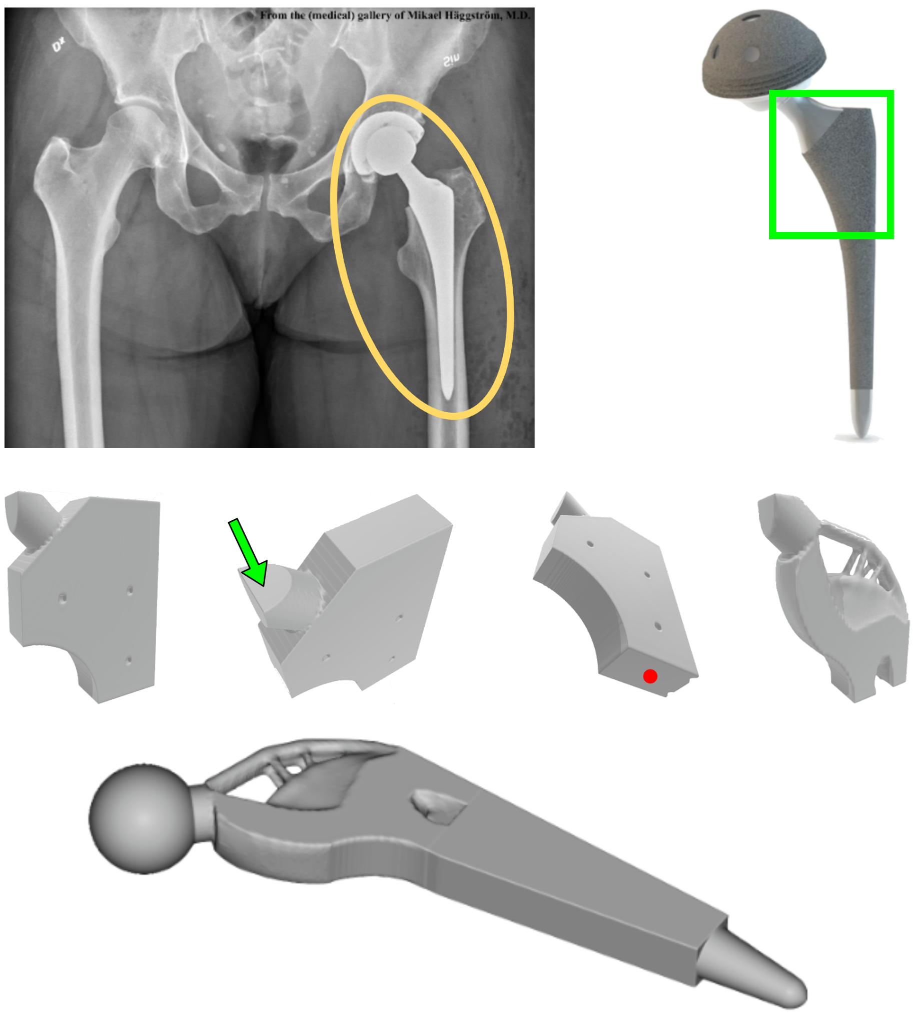


Figure 5-9: Topology optimization of the hip implant.

This area serves as an optimization domain D in the optimization algorithm with volume restriction equal to 45% of the total volume of the domain. However, the domain D has several features. The material from the circular area of the domain near the bottom left edge was subtracted. The upper left edge area has material only in the cylindrical area. These restrictions of the geometry were preserved during optimization.

R-functions were used to incorporate the restrictions into the domain. The optimized body can be written in terms of Boolean operations as

$$\Omega = ((\Omega_o \backslash \Omega_c) \backslash \Omega_t) \cup \Omega_s$$

where Ω_o is a free-form body defined by Equation 5.11, Ω_c is a circular area near the bottom left edge, Ω_t is a triangular area near the upper left edge and Ω_s is cylindrical steam. Similarly, the defined function of the result body can be written as:

$$f(x,y,z) = ((f_o(x,y,z) \backslash f_c(x,y,z)) \backslash f_t(x,y,z)) \vee f_s(x,y,z)$$

where $f_0(x,y,z), f_c(x,y,z), f_t(x,y,z)$ and $f_s(x,y,z)$ are defining functions of corresponding bodies, $\backslash$ and $\vee$ are R-functions for subtraction and union operations respectively. The optimization algorithm modifies only the free-form part $f_o(x,y,z)$ of the defining function $f(x,y,z)$ in this optimization case. The MATLAB implementation of the optimization algorithm (see `Examples \3D_femur_optimizer.mlx` in Popov [2022a]) uses *min* and *max* versions of R-functions for Boolean union and subtraction.

5.6 Parameter optimization

The second option of non free-form optimization is a parameter optimization. It works with parameters of FRep objects to find the optimal configuration with respect to a considered problem. Parameter optimization has much less freedom in topological changes in comparison with topology optimization. Thus, more straightforward algorithms can be applied for its solution.

This dissertation proposes using the parameter optimization algorithm based on the gradient descent method listed in Algorithm 3. Let us consider how it works on the example. The optimization task is shown in Figure 5-2(a) left. The following parameter optimization task is proposed. A circular hole of the constant radius

R inside the domain is added. The goal is to optimize its position inside the domain with respect to the compliance minimization problem. Figure 5-10 shows this problem configuration.

Algorithm 3: The parameter optimization algorithm for FRep objects

```
Step 1. Define the number of grid elements in the domain $D$ to perform
  FEM analysis;
Step 2. Initialize the FRep defining function;
Step 3. Define boundary conditions for FEM analysis;
Step 4. Initialize parameters of the optimization loop;
Step 5. Perform FEM analysis of the domain;
Step 6. If the algorithm converges, then quit, else go to Step 8;
Step 7. Find the descent direction of parameters for the optimization
  problem;
Step 8. Update parameters according to the descent direction;
Step 9. Apply constraints defined for parameters and go to Step 5.
```

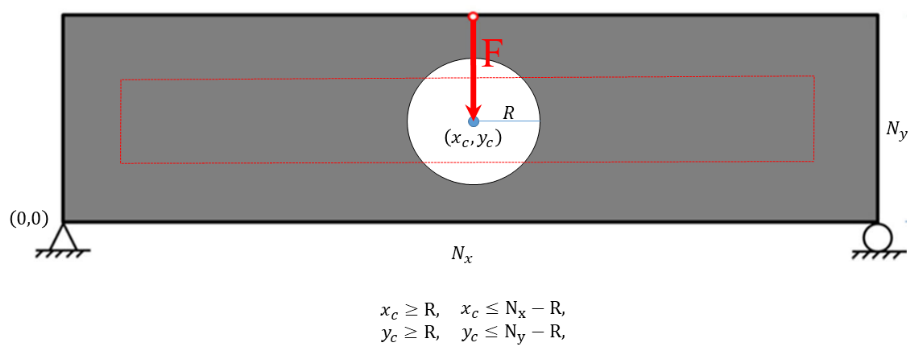

F
R
(x_c, y_c)
N_y
(0,0)
N_x
$x_c \geq$ R, $x_c \leq$ N$_x$ − R,
$y_c \geq$ R, $y_c \leq$ N$_y$ − R,

Figure 5-10: One hole parameter optimization problem statement.

The red dashed rectangular area in Figure 5-10 defines the region of allowed positions for the hole center. That means that the hole escape from the design domain is restricted.

One can specify steps of Algorithm 3 referring the considered example. It use the same FEM with ersatz material (see Section 2.3) model to perform mechanical simulations as used for the topology optimization. It is independent from the shape of the optimized body and do not require re-meshing. At the same time, this approach provide us with reliable results as one can see from Section 5.3.

`Step 8` includes a search of descent direction for parameters variation. The proposed implementation of the algorithm use numerical differentiation of the objective function for this purpose. There are two partial derivatives needed to be calculated for the example in Figure 5-10. They are responsible for vertical and horizontal movements of the hole location.

This differentiation gives us the decent direction with respect to the compliance energy. The update of parameters happening at `Step 9` depends on the obtained direction. In the proposed algorithm the constant gradient descent step γ is used. It is equal to 10^{-5} for the considered example shown in Figure 5-10.

`Step 10` of the algorithm prevents parameters from violation of given limitations. It is the hole escaping for the considered example. Values of the hole coordinate are limited with R from the bottom and with $N_i - R$ from the top. N_i are two dimensions of the domain D. It is easy to keep coordinates within this range by forced setting them equal to R or $N_i - R$ when they violate the bottom or top limitations respectively.

Figure 5-11 shows the results of the example optimization. It has three cases of initial designs. They led for different optimization results where the first one is the closest to the global optimum. However, all of them can be considered as good enough because their values of objective functions differ just in the third significant figure. It is possible to run the algorithm for different initial designs and choose the best optimized option if it is crucial to obtain the best solution.

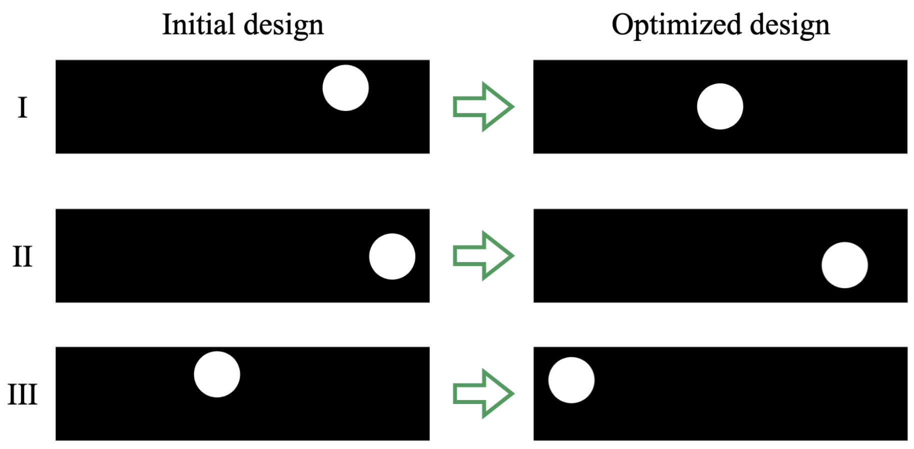


Figure 5-11: One hole parameter optimization results for tree different initial designs.

Figure 5-12 helps us to understand why the optimization algorithm can stuck at these configurations of the FRep object. It shows the surface in 3D generated by values of the objective function depending on coordinates of the hole. As you can see from the picture there are thee regions with optima. Two of them are local in the left and right sides of the domain and the third one is the global optimum in the center region.

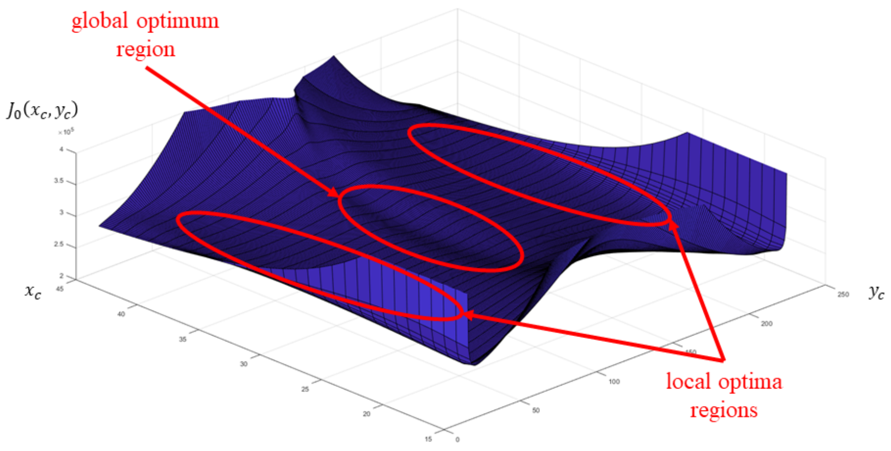


Figure 5-12: The plot of the objective function values for the one hole parameter optimization problem.

It is difficult to propose more specific and formal algorithm for parameter optimization than Algorithm 3. For example, `Step 10` has constraints processing which strictly depend on the formulation of the optimization problem. Moreover, even the simple case considered above has non-convex objective function. Therefore, each specific parameter optimization task or each class of parameter optimization task requires its own approach to solution. For example, one can use coordinate descent or random coordinate descent when the number of parameters is big.

Figure 5-13 shows more complex example of parameter optimization. The domain D has several holes with varying coordinates of their centers and radii equal to fixed value R. In this case, one need to control not only escaping holes from the domain but also prevent their merging.

The formal definition of the optimization algorithm is shown in Algorithm 4. It uses acceptable function error ε_J and position error ε_x for stopping criteria. The while loop condition exploits them in the algorithm. However, this condition can be updated with the requirement that it should be violated several times, e.g. 10. It allows the algorithm to overcome regions where the objective function is "flat".

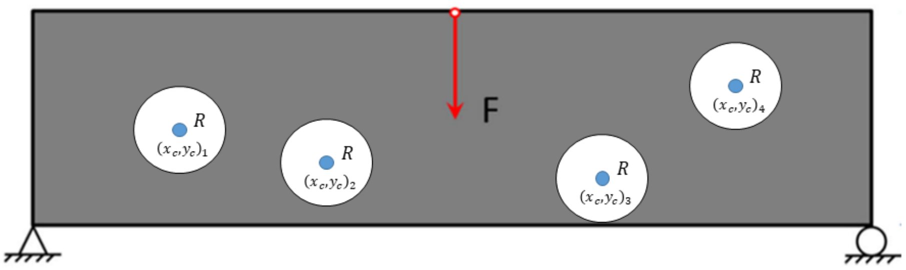


Figure 5-13: Parameter optimization problem statement for the case of several holes.

Algorithm 4: Optimization placing of several holes into the rectangular domain

```
$k \leftarrow 0$ ;                                        /* iteration number */
$t \leftarrow 0$;
$\gamma \leftarrow 10^{-5}$ ;                             /* constant updating step */
$\varepsilon_J \leftarrow 10^{-8}$;
$\varepsilon_x \leftarrow 10^{-3}$ ;          /* $10^{-5}$, $10^{-8}$ and $10^{-3}$ are example values */
$\forall i = 1, \ldots, m$: randomly generate $(x_c, y_c)_i^k$;
$\forall i = 1, \ldots, m$: $(x_c, y_c)_i^{k+1} \leftarrow (0, 0)$;
while $\forall i = 1, \ldots, m$: $\left\| \left( \frac{\Delta J_0}{\Delta x_i}, \frac{\Delta J_0}{\Delta y_i} \right)_i^k \right\|_2 \geq \varepsilon_J$ and
 $\left\| (x_c, y_c)_i^{k+1} - (x_c, y_c)_i^k \right\|_2 \geq \varepsilon_x$ do
    $\forall i = 1, \ldots, m$:
      $(x_c, y_c)_i^k \leftarrow \Big( min\big(R, max\,(x_c, N_x - R)\big), min\big(R, max\,(y_c, N_y - R)\big) \Big)_i^k$;
    forall $i \neq j$ do
        $t \leftarrow \left\| (x_c, y_c)_i^k - (x_c, y_c)_j^k \right\|_2$;
        if $t < 2R$ then
            if $R < x_c^j + 2R\frac{x_c^i - x_c^j}{t} < N_x - R$ and $R < y_c^j + 2R\frac{y_c^i - y_c^j}{t} < N_y - R$
             then
                $(x_c, y_c)_i^k \leftarrow (x_c, y_c)_j^k + 2R\frac{(x_c,y_c)_i^k - (x_c,y_c)_j^k}{t}$;
            else
                $(x_c, y_c)_j^k \leftarrow (x_c, y_c)_i^k + 2R\frac{(x_c,y_c)_j^k - (x_c,y_c)_i^k}{t}$;
    $\forall i = 1, \ldots, m$: $(x_c, y_c)_i^{k+1} \leftarrow (x_c, y_c)_i^k - \gamma \left( \frac{\Delta J_0}{\Delta x_i}, \frac{\Delta J_0}{\Delta y_i} \right)_i^k$;
    $k \leftarrow k + 1$;
```

The variable t serves as an indicator of holes merging. A relaxed routing was applied to prevent it. Each iteration of the optimization loop checks the merging of holes pairwise and spread them. Thus, there are iterations when the proposed design violates the merging constraint. However, the second loop condition prevents optimization finish while the holes configuration is not stable.

Figure 5-14 shows three examples of the proposed algorithm results. One can see that the difference in values of the objective function appears in the second significant figure. Moreover, more symmetric initial design resulted with better solution. Thus, for more complex parameter optimization task one can see bigger difference between global and local optima. It means that for such cases it is more attractive to run the optimization algorithm several times with different initial guesses. One more interesting thing in Figure 5-14 is that the third case has similar pattern in the holes distribution as voids of the optimized free-form body (see Figure 5-3).

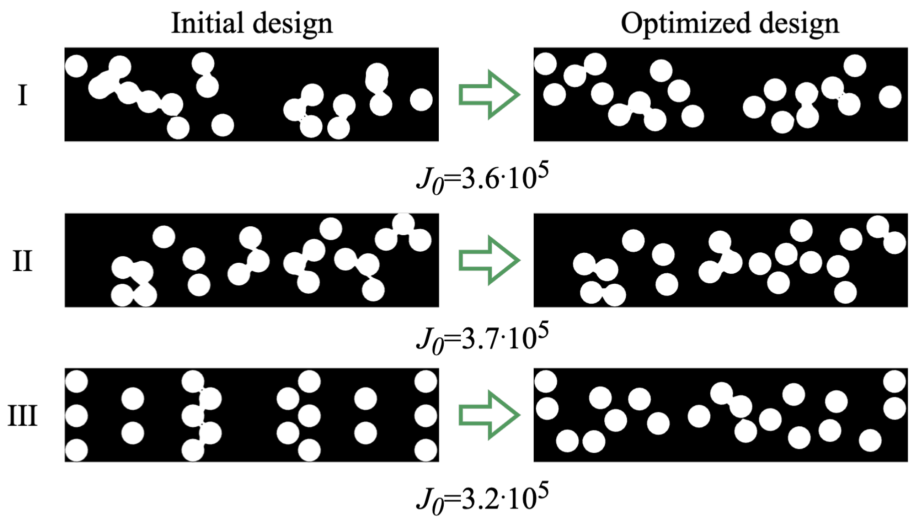


Figure 5-14: Parameter optimization results for the case of several holes.

5.7 Structural optimization

Structural optimization means the process that includes both topology and parametric optimizations. One can construct a structural optimization algorithm by combining the topology optimization algorithm with shape constraints listed in Section 5.5 and the parameter optimization algorithm proposed in Section 5.6. The result is shown in Algorithm 5.

As in the case of optimization with constraints (see Section 5.5), an FRep defining function used in this algorithm consists of two parts. One can similarly write it as $f = f(x, y, f_o)$ or $f = f(x, y, z, f_o)$. It includes a free-form function f_o those coeffi-

Algorithm 5: The structural optimization algorithm

```
Step 1. Define the number of grid elements in the domain $D$;
Step 2. Initialize the FRep defining function;
Step 3. Define boundary conditions for FEM analysis;
Step 4. Initialize parameters of the optimization loop;
Step 5. Perform FEM analysis of the domain;
Step 6. If the algorithm converges, then quit, else go to Step 7;
Step 7. Update coefficients of the free-form function;
Step 8. Find the descent direction of parameters for the optimization
  problem;
Step 9. Update parameters according to the descent direction;
Step 10. Apply constraints defined for parameters and go to Step 5.
```

cients evolve during the topology optimization routine applied at `Step 7`. However, the rest part of the defining function has changing parameters.

Figure 5-15 shows results of structural optimization for tree cases of initial design. These results are solutions of the problem shown in Figure 5-2(a) left. However, the ring was added into this design and considered its location as a parameter. Values of the objective function for these cases differ in the second significant digit as in the case of parameter optimization. Therefore, the algorithm again faces with the problem of local minima and one need to run the optimization algorithm several times to obtain a better result.

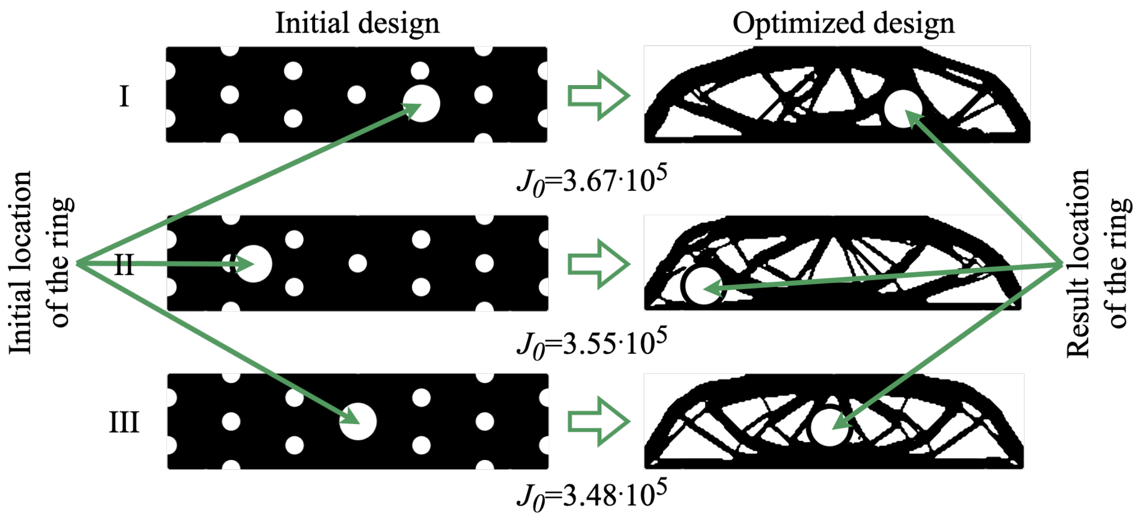


Figure 5-15: Results of structural optimization.

5.8 Multimaterial optimization of FRep objects

The topology optimization algorithm described in Section 5.1 can be applied for the case of several materials. One can apply it for each of material separately and set the values of volume constraints to obtain a desired proportion in the result material distribution.

However, one need to somehow prevent regions from appearing with a mixture of material. One material can be chosen as a dominant and it will occupy these common areas.

Two free-form defining functions f_1 and f_2 in the form of Equation 5.3 or Equation 5.11 are suitable for describing portions of the body made of two different material. Then, one can define f_1 as the dominant one. In this case, the optimization algorithm includes the resolving step $f_2 = f_2 \setminus f_1$. The subtraction operation used here can be chosen as some of R-functions proposed in Subsection 2.1.2. The use of its min version, i.e. $f_2 = min(f_2, -f_1)$ preserves the efforts made to restrict the unbound growth of defining functions.

The same strategy can be applied for bigger number of materials. However, in this case one need to define a material hierarchy instead of a single dominant material. The same routine of material subtraction can be applied for several materials in the consequent manner. Thus, the proposed topology optimization algorithm can be stated as listed in Algorithm 6.

Algorithm 6: The proposed topology optimization algorithm

```
Step 1. Define the number of grid elements in the domain $D$;
Step 2. Initialize coefficients of the free-form functions for each material;
Step 3. Define materials and their hierarchy;
Step 4. Define boundary conditions for FEM analysis;
Step 5. Initialize parameters of the optimization loop;
Step 6. Perform FEM analysis of the domain;
Step 7. If the algorithm converges, then quit, else update coefficients of the
  free-form functions, perform the common areas resolving and go to Step 6.
```

The term material means concrete values of the Young's modulus E and the Poisson's ratio μ for the considered compliance minimization problem. Therefore, `Step 3` of the proposed algorithm sets these values and the consequence which be used to apply pairwise subtraction operation.

Figure 5-16 shows the results of an example multimaterial optimization. It was performed for the problem shown in Figure 5-2(a) left. The total volume constraint used here is 50% of the initial domain volume. Each of the material must occupy 25% of the domain in this setting. Both materials have the same Poisson's ratio $\mu = 0.3$ and Young's modulus $E_1 = 1$ and $E_2 = 3$. The first material is shown in red color and the second one – in the blue color. Tiny white contours between two materials in this picture do not show the real material distribution but just issues of drawing.

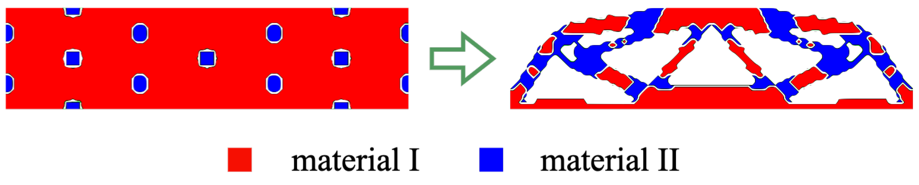


Figure 5-16: Results of multimaterial optimization.

The proposed algorithm does not guarantee the solution of the problem in the sense of the global optimum. It results with local solutions as algorithms considered before. Moreover, it requires more computation time because of processing of several materials. It can be updated similarly as the previously considered algorithm to capture 3D, shape constraints, parametric and structural optimization cases.

Chapter 6

Slicing of FRep objects

This chapter delivers the results on developing CAM component of the proposed system. These results were published in papers [Popov et al., 2020b] and [Maltsev et al., 2021].

The first paper studies contouring algorithms that can be applied for AM. Contouring is the most time expensive step of generating CNC program for FRep model. However, it proposed efficient algorithms for solving this task and recommendations of their choice.

The second paper considers the question, how one can use specific forms of FRep defining functions to speed up contouring. This paper ([Maltsev et al., 2021]) proposed efficient methods of CNC program generation for models with cell repetition and, more generally, for models operating with huge number of defined knot points. The proposed algorithms work well with free-form functions (Equation 5.3 and Equation 5.11) used for topology optimization.

6.1 Adaptive contouring algorithms

The 2.5 representation of models or slices was used to generate the toolpath of a 3D printer. This representation is based on a layer-by-layer description of a model, where each layer consists of one or several disjoint contours. The number of layers depends on the resolution of the machine and on the level thickness.

Contouring the geometric models depends on the geometry type. One can formally define contours of FRep models on level $z = z_0$ as the 2D point set $H = \{(x, y) \mid f(x, y, z_0) = 0\}$. In our text, 2D objects are built of contours on level z_0 as $f_{z_0}(x, y) = f(x, y, z_0)$. It can be seen that the contour set for the given layer is a zero-level set of a functionally represented model in 2D, which is called an implicit curve in some works. Additionally, note that each level in this formal approach has zero thickness, and intra-level connectivity is assumed by the AM process.

The resolution of the machine in both the XY and Z directions is important in this process because it affects the quality of the final result. The resolution in the Z direction defines how many slices have to be produced and sent to the machine to define the layers. It should be noted that there are methods with a variable z-step [Attene et al., 2018], but this work only considers methods with a constant z-step. The resolution in XY, on the other hand, defines the quality of the contours in each layer, which in functionally represented geometry means the quality of the approximation. Higher approximation quality requires longer processing times for the approximation process, which very often makes the contouring process time-consuming.

The topology of the implicit curve defined by the function $f_{z_0}(x, y) = 0$ depends on the function itself, and for models with microstructures, this curve can contain hundreds, if not thousands, of disjoint contours per layer. The goal of the proposed system is to efficiently approximate the set H for the given resolution of the target AM hardware. Moreover, it shall ensure that all contours are included in the toolpath for the current layer.

6.1.1 Conventional contour extraction for AM

Conventional methods for extracting the 2D contours of a model defined in an implicit form can be classified as follows:

1. Exhaustive enumeration or contour extraction on a regular grid [Lorensen and Cline, 1987], [Theußl et al., 2001], [Carr et al., 2003].

2. Adaptive subdivision of the space on adjacent cells [Herzen and Barr, 1987], [Duff, 1992], [Stolte and Kaufman, 1998].

3. Surface tracking or numerical continuation [Wyvill et al., 2005], [Rangan et al., 1997], [Levinski and Sourin, 2002].

The methods in the third category essentially perform contour extraction based on moving according to implicit surface using piecewise-linear methods or predictor-corrector methods with incremental partitioning [Bloomenthal, 1988]. Continuity-based methods tend to extract contours by taking into account the tangent and curvature of implicit curves. It takes a point on the implicit curve as the starting point and then moves forward at a certain distance along the tangent of the implicit curve at the starting point to predict the position of the new point. Then, the new point is corrected for closing to the curve along the direction of the gradient of the field defined by the implicit function. In the case of a layer containing a very large number of contours, it requires finding a seed point on every single component, which is a tedious task that requires solving the problem of component analysis for an implicitly defined curve [Bloomenthal and Wyvill, 1997]. Because of the described limitation, this method will be excluded from discussion. Below, the thesis discussed the conventional methods from the first two categories.

Contour extraction on a regular grid

The methods of contour extraction on a regular grid, mainly based on the marching squares algorithm [Lorensen and Cline, 1987], are widely used in many applications dealing with implicitly defined curves and surfaces.

The basic idea of this approach is to split the contouring domain into a regular grid, then check the edges of these cells for a sign change in the vertices and find the points of the target curve or surface. There are cells of 16 types (see Figure 6-1) defined by the sign values in the cell vertices. Fourteen of these types mean that a cell contains a piece of the resulting contour. In the implementation of this method, the traversal starts from the bottom left cell. The traversing direction is from bottom to top and left to right.

The algorithm operates on objects such as line segments and polylines. A polyline is an oriented chain of segments. The result of the algorithm is a set of polylines. These polylines are closed for correct FRep models and a proper bounding box.

FRep model is correct if it is defined by a continuous function. The bounding box here is a rectangular domain where the regular grid is constructed.

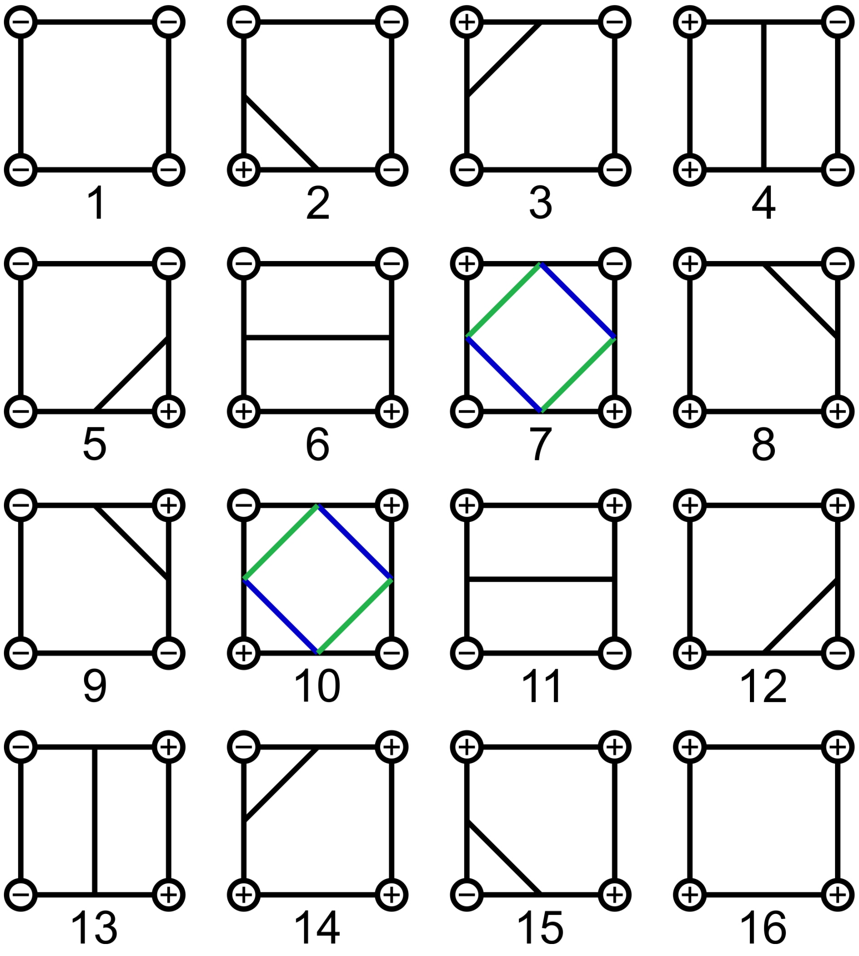


Figure 6-1: Look-up table for marching squares.

Therefore, the process of contour creation includes two tasks: to process cells and to connect resulting segments properly. However, two types of cells, 7 and 10, produce ambiguous configurations, which need to be resolved separately. There are several ways to resolve this issue. For example, in [Uchibori, 2004], ambiguity is avoided by local subdivision of ambiguous cells. In this case, however, significant extra time can be spent on this splitting process, and the resulting grid is not regular, which can potentially result in a broken contour and the process going beyond the calculation tolerance.

The second approach to the problem solving is to calculate the additional value of the function at the centre of a cell [de Araújo et al., 2015], [Wyvill et al., 1986]. An example of resolving ambiguity inside cell number 7 is illustrated in Figure 6-2.

If the function value of the cell centre is positive, then this cell has the same type as in Figure 6-2(a); otherwise, the cell has the same type as in Figure 6-2(b). This method relies on scalar field regularity and can lead to an incorrect conclusion in the case that a field is non-symmetrical with respect to the cell in question.

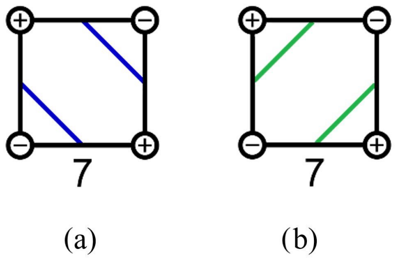


Figure 6-2: Variants of ambiguity resolution inside cell number 7.

In our method, the defining function $f_{z_0}(x,y)$ is approximated with a bilinear function:

$$h(x,y) = f_{z_0}(x_1,y_1)\frac{x_2-x}{x_2-x_1} + f_{z_0}(x_2,y_1)\frac{x-x_1}{x_2-x_1}\frac{y_2-y}{y_2-y_1} + \\ + f_{z_0}(x_1,y_2)\frac{x_2-x}{x_2-x_1} + f_{z_0}(x_2,y_2)\frac{x-x_1}{x_2-x_1}\frac{y-y_1}{y_2-y_1}$$

for the considered cell (see Figure 6-3). Here, (x_1,y_1) is the lower left and (x_2,y_2) is the upper right vertices of the cell. Similar approach is described in [Pasko et al., 1988] for 3D surfaces. The proposed approximation resolves the ambiguity. It works better for smooth defining functions. Moreover, this method does not require refinement of the grid.

The function $h(x,y)$ is an implicit description of a hyperbola. Its centre has the following coordinates:

$$x_c = \frac{x_2 f_{z_0}(x_1,y_1) + x_1 f_{z_0}(x_2,y_2) - x_1 f_{z_0}(x_2,y_1) - x_2 f_{z_0}(x_1,y_2)}{f_{z_0}(x_2,y_2) + f_{z_0}(x_1,y_1) - f_{z_0}(x_1,y_2) - f_{z_0}(x_2,y_1)}, \tag{6.1}$$

$$y_c = \frac{y_2 f_{z_0}(x_1,y_1) + y_1 f_{z_0}(x_2,y_2) - y_1 f_{z_0}(x_1,y_2) - y_2 f_{z_0}(x_2,y_1)}{f_{z_0}(x_2,y_2) + f_{z_0}(x_1,y_1) - f_{z_0}(x_1,y_2) - f_{z_0}(x_2,y_1)}. \tag{6.2}$$

In addition, it can be seen that for ambiguous cells with $x_1 \leq x_c \leq x_2$ and

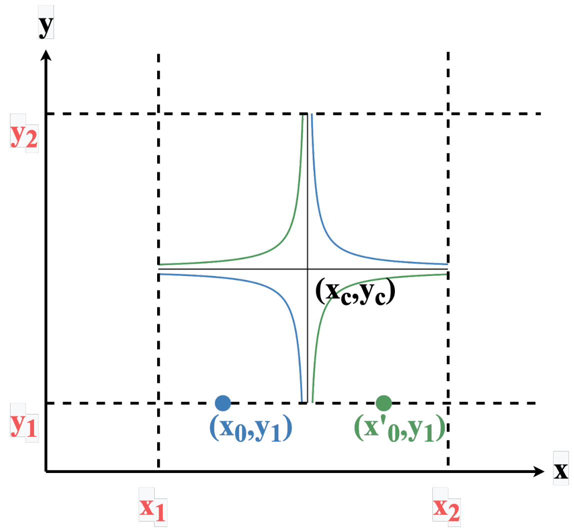


Figure 6-3: Bilinear approximation of the defining function.

$y_1 \leq y_c \leq y_2$, two possible configurations of the hyperbola are possible. For example, for $y = y_1$, one can obtain $x_0 \leq x_c$ or $x_c \leq x'_0$, where the value of the defining function is zero. The first option leads to the "blue" cell configuration and the second to the "green" configuration in Figure 6-2.

It should be noted that this approach can also produce the wrong result in the meaning of the true shape of the model. Nevertheless, it does not require to calculate function values at any extra point.

In the context of AM, the parameters for a regular grid are defined by the target precision of the machine. For example, the precision r defines the minimal length of the cell in the X or Y direction, and for fused filament fabrication, this value is 30-300 μm. This means r is the minimal distance that can guarantee that two different points will not be merged.

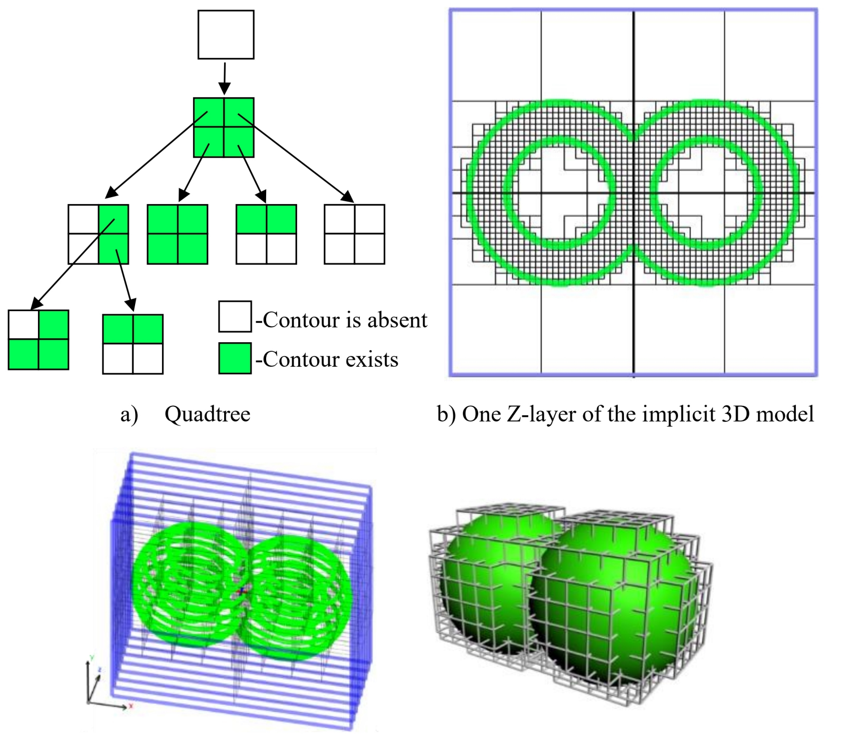
Contour extraction using adaptive subdivision of the space

It is clear that for very small resolution, conventional methods are not efficient because of the necessity to calculate the value of the defining function at very large number of points. To avoid this limitation, an adaptive subdivision is used. The main idea is to increase the resolution locally, i.e., to localize the areas where $f(x, y, z) = 0$ and subdivide these areas up to the required precision to optimize the time by decreasing the number of calculations. The contour extraction for one

Z-layer is performed using a marching squares algorithm with adaptive subdivision using the quadtree [Bloomenthal, 1988].

The quadtree is built from the root node, which is the bounding box of the implicit curve. Then, it is divided into four equal regions, which form the child nodes. Every child node can be recursively divided further.

The process of contour extraction using the quadtree for one Z-layer and for the whole model is shown in Figure 6-4(a-d). The equations for the functionally defined model for these pictures are presented in Appendix B.



a) Quadtree b) One Z-layer of the implicit 3D model

c) All layers of implicit 3D model d) Visualization of implicit 3D model

Figure 6-4: The process of contour extraction using the quadtree.

The simplest approach to build an adaptive subdivision is to use the marching squares algorithm with rough resolution to understand the topology of the implicit curve. The marching squares algorithm checks the sign differences in the corners of the processed cell for contour detection. However, this criterion is not robust and can miss several cases. Figure 6-5 shows examples of these cases [Bloomenthal, 1988], [Kalra and Barr, 1989].

Another method to identify the cells that need to be subdivided is to use In-

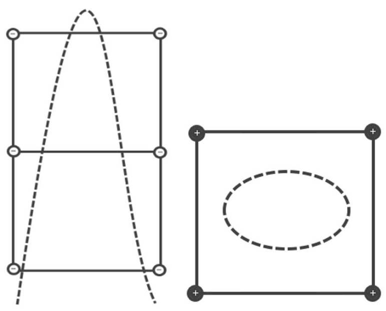


Figure 6-5: Errors of the marching squares algorithm.

terval Arithmetic (IA). IA is an extension of real arithmetic defined on the set of real intervals [Moore, 1966], [Sunaga, 2009], [Warmus, 1956]. In IA, every quantity is represented by an interval of real numbers. An interval $[a, b]$ is a set of $x \in \mathbb{R}$ such that $a \leq x \leq b$. During functions processing in IA, each quantity is replaced by its interval extension, and all computations are executed on intervals. There are extensions of the usual operations $(+, -, \times, /)$ for intervals. These operations guarantee that each computed interval includes the whole range of function values on the defined argument range. A more detailed description can be found in Appendix A.

IA allows us to estimate the upper and lower bounds of the range of the function values for each cell of the quadtree. The axis-aligned boundaries of processed quadtree cells are the intervals used as function arguments. The result of the computation of the defining function in IA is also an interval. If the upper and lower bounds of the interval are on opposite sides of zero, then the defining function changes signs inside the interval. This means that the processed quadtree cell may contain contours of the curve and should be divided [Bühler, 2002], [Duff, 1992]. An example of an interval estimation for a function with one variable is shown in Figure 6-6.

The green line in Figure 6-6 is the contour $f(x, y, z = const) = 0$. The estimation of the function value $f(x, y, z = const)$ in the quadtree cell with intervals X and Y is the calculated interval of $F(X, Y, Z = const)$, where $F(X, Y, Z = const)$ is a natural interval extension for the function $f(x, y, z = const)$. The natural interval extension for a function is the interval extension with intervals as function arguments and the interval arithmetic operations performed on them.

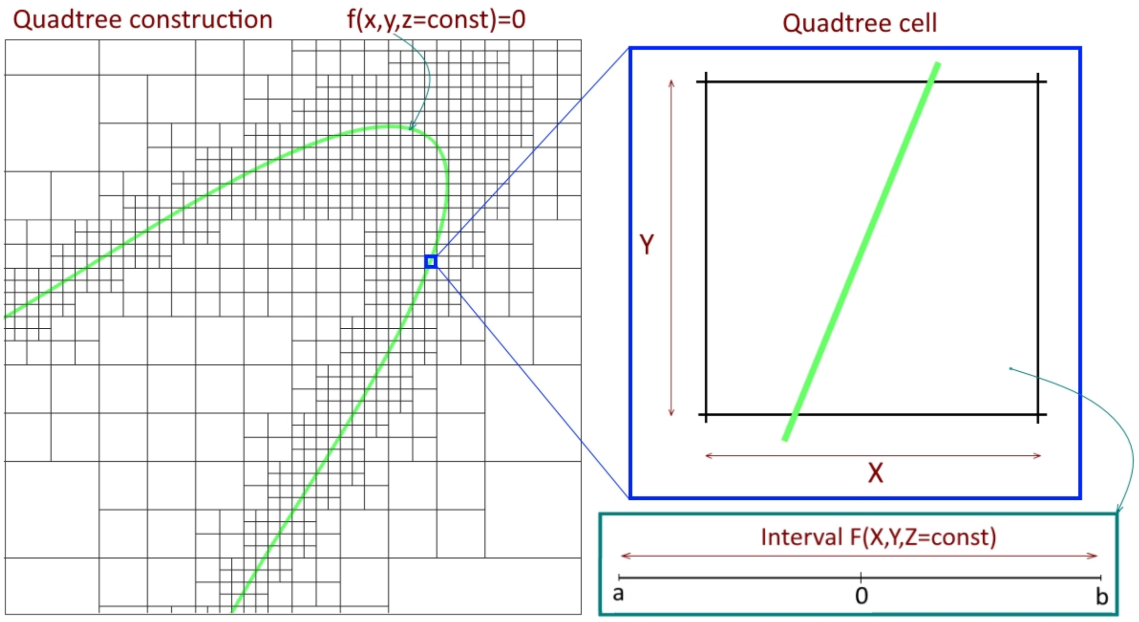


Figure 6-6: An example of an interval estimation.

The model with union of spheres contoured using the IA criterion is shown in Figure 6-7(b). However, using IA in contour extraction results in the overestimation of the function interval values, especially if the function includes a large number of nonlinear operations. As a result, it leads to an increase in the number of calculations [Fryazinov et al., 2010].

6.1.2 New approach to contour extraction for AM

IA is used for contouring of FRep [Song et al., 2018], [Mochizuki, 2004]. However, using the IA approach with the adaptive subdivision algorithm has the drawbacks mentioned above. Thus, the algorithm applies an improved version of IA, Affine Arithmetic (AA). Its main purpose is to improve the accuracy of the interval estimation in comparison to IA.

The main ideas of AA can be found in Stolfi and Figueeirdo's introduction [Stolfi and de Figueiredo, 2003]. In AA, all quantities are represented in affine form as first-degree polynomials with coefficients and symbols for unknown real-valued variables. These variables are independent and are called noise symbols. The affine forms are constructed for each quantity and operation in the defining function. Then, the computations are performed with the affine forms of the defining function. This

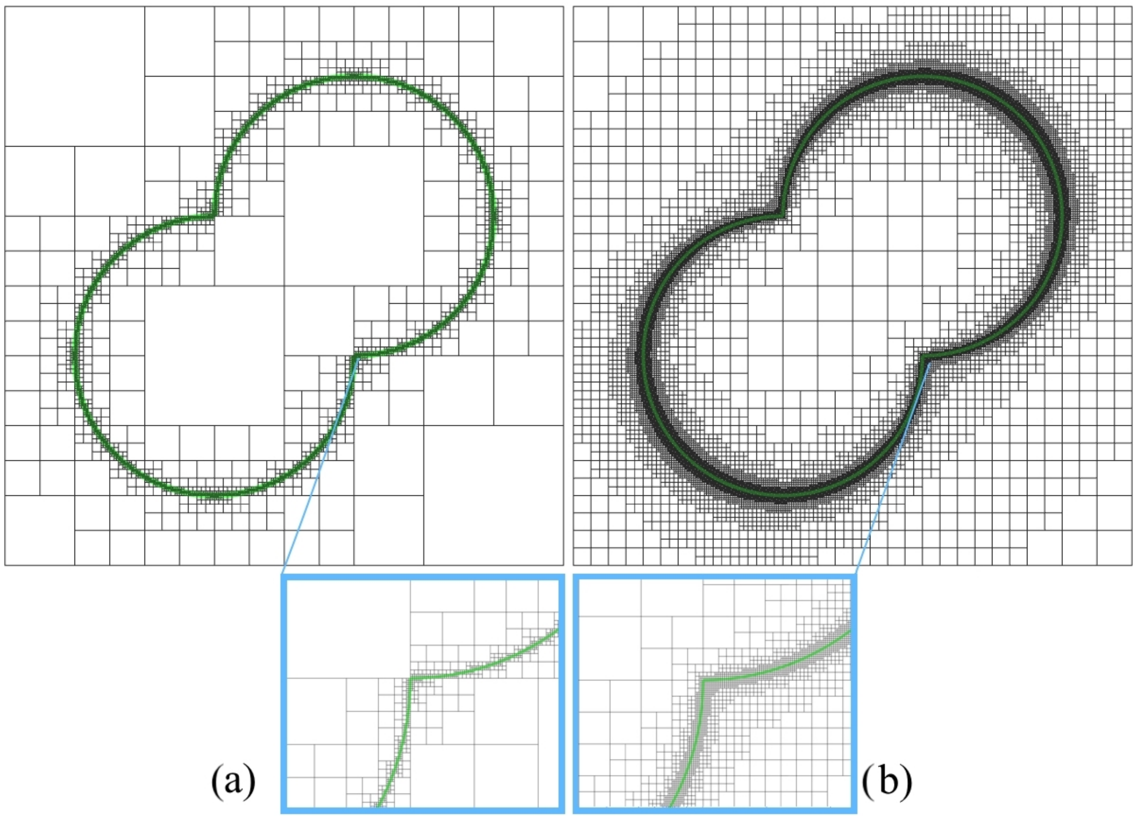


Figure 6-7: The quadtree construction and the contouring process for one layer of the sphere-union model using: (a) the affine arithmetic criterion, (b) the interval arithmetic criterion.

allows us to obtain an interval estimation of the defining function with more accurate ranges than by using interval arithmetic.

AA was revised in [Vu et al., 2004], and several approximations for non-affine operations were improved. The more detailed description of AA can be found in Appendix A. Many applicable approximations were collected in [Rump and Kashiwagi, 2015]. Furthermore, AA was used for the purposes of performing spatial enumeration of implicit surfaces (n is the number of arguments of the defining function, i.e., the dimension [Fryazinov et al., 2010]).

The revised affine form of a real-valued quantity $\hat{x}$ consists of two parts. There are the standard affine part of the length n and the interval part:

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i + e_x[-1, 1], e_x \geq 0,$$

where the x_is are finite real numbers and ε_i are unknown real-valued variables located within the interval U = [-1;1]. The coefficient x_0 is called the central value of the affine form $\hat{x}$, and the coefficients x_i are called the partial deviations. ε_i are called the noise symbols. $e_x[-1,1]$ is a cumulative error, which represents the maximum absolute error of non-affine operations. One of the main constraints of pure AA is that noise symbols increase dramatically during computations. $e_x[-1,1]$ accumulates the noise symbols that are present in pure AA and are not dependent on the input values. This means that the length of the revised affine form does not exceed the number of input variables during computation. In the interrogation methods for contouring the implicit curve, three coordinates of 3D space are used as input variables.

If two quantities x and y are represented in revised affine forms:

$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + ... + x_n\varepsilon_n + e_x[-1,1], \tag{6.3}$$

$$\hat{y} = y_0 + y_1\varepsilon_1 + y_2\varepsilon_2 + ... + y_n\varepsilon_n + e_x[-1,1], \tag{6.4}$$

then, the affine operation $f(\hat{x},\hat{y}) \equiv \alpha\hat{x} + \beta\hat{y} + \gamma$ using the revised affine form can be written as follows:

$$f(\hat{x},\hat{y}) = (\alpha x_0 + \beta y_0 + \gamma) + \sum_{i=1}^{n}(\alpha x_i + \beta y_i)\varepsilon_i + (|\alpha|e_x + |\beta|e_y)[-1,1], (\alpha,\beta,\gamma \in \mathbb{R}), \tag{6.5}$$

where α, β, γ are real-valued coefficients; x_0, y_0 are the central values of the revised affine forms $\hat{x}$ and $\hat{y}$, respectively; the coefficients x_i are the partial deviations of the revised affine forms; the ε_is are the noise symbols; and e_x, e_y are cumulative errors.

There is a special tight form of the product of two revised affine forms $\hat{x}$ and $\hat{y}$ of length n:

$$\hat{x} * \hat{y} = (x_0y_0 + \frac{1}{2}\sum_{i=1}^{n} x_iy_i) + \sum_{i=1}^{n}(x_0y_i + x_iy_0)\varepsilon_i + e_{xy}[-1,1], \tag{6.6}$$

where

$$e_{xy} = e_xe_y + e_y(|x_0|+u) + e_x(|y_0|+v) + uv - \frac{1}{2}\sum_{i=1}^{n}|x_iy_i|, u = \sum_{i=1}^{n}|x_i|\ v = \sum_{i=1}^{n}|y_i|. \tag{6.7}$$

In this work, the revised AA was implemented to calculate the adaptation criterion for quadtree space division. This means that for our defining function *f*, its AA version is needed. It can be obtained by replacing all non-linear operations by their AA counterparts with the techniques explained in the relevant literature discussed above. To define whether a cell needs to be subdivided, the algorithm calculates its AA function and analyse the interval containing the zero value, skipping the cells that do not contain it. On the last level of subdivision, the algorithm calculates the defining function value and process the cells in the same way as marching squares does. The result of this algorithm is a polyline, which is used as a toolpath in our direct fabrication process for the given layer. The resulting algorithm is listed in Algorithm 7.

Algorithm 7: Contouring with adaptation criteria

```
Step 1. $BuildQuadTree(x_0, y_0, z_0, BoundingBoxLength)$;
Step 2. Create the topology of the curve;
Step 3. Calculate values of $f(x, y, z)$ on edges;
Step 4. Create a polyline using marching squares;
```

Procedure BuildQuadTree($x, y, z, CellLength$)

```
if $CellLength > Precision$ then
    if $AdaptationCriterion(x, y, z, CellLength)$ contains 0 then
        $CellLength \leftarrow CellLength/2$;
        $BuildQuadTree(x, y, z, CellLength)$ ;
        $BuildQuadTree(x + CellLength, y, z, CellLength)$;
        $BuildQuadTree(x, y + CellLength, z, CellLength)$;
        $BuildQuadTree(x + CellLength, y + CellLength, z, CellLength)$;
    else
        Reject Cell
else
    Add a cell as a quadtree leaf;
```

`Step 2` of this algorithm uses the connected component labeling algorithm [Shapiro, 1996]. It exploits the bisection method for extracting polyline edges at `Step 4`. `AdaptationCriterion(x,y,z,CellLength)` here is AA or IA extension of the FRep defining function Equation 2.1.

The model with union of spheres contoured using the AA criterion is shown in

Figure 6-7(a). Here the algorithm obtained 29781 cells for IA in 250 ms and 11021 cells for AA in 100 ms for a given resolution of 0.01 mm. IA and AA provide us with the same contours, although AA works 2.5 times faster and reduces the number of explored cells for this model.

6.1.3 Experimental results of contouring algorithms

Contouring algorithms were compared on models of different complexity. Models were prepared with our approach for direct fabrication, i.e., sliced by using contouring methods, and every contour defined a path for 3D printing hardware. Table 6.1 shows different models with respect to the dimensions of their fabricated versions.

Algorithms were implemented and tested on a PC with an Intel Core i5-8250U CPU @1.60 GHz, 1.80GHz, 8 Gb RAM, with multi-threading through OpenMP OpenMP [2022]. The template-based Boost library Melquiond et al. [2006] was used for IA and the authors' implementation of the revised AA [Fryazinov et al., 2010].

Algebraic surfaces (see Figure 6-8) along with more complex non-algebraic surfaces (see Figure 6-9) with procedural microstructures and set-theoretic operations, which are based on R-functions (see Table 2.1), were used. The defining functions of the models can be found in Appendix B.

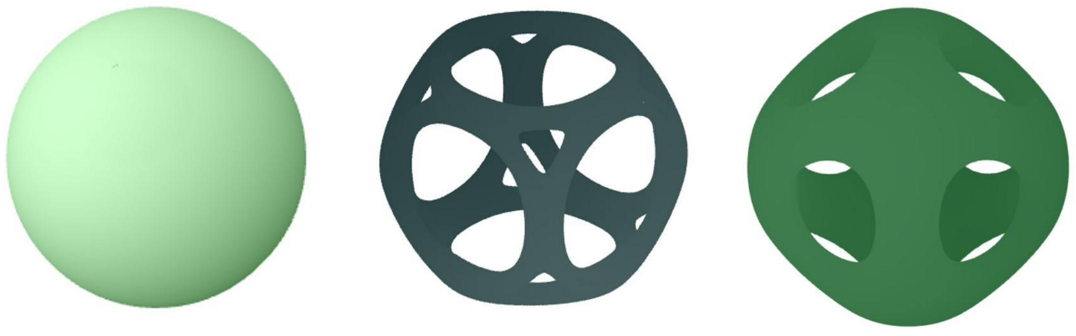


Figure 6-8: Algebraic surfaces.

To test contouring with various resolutions for the models presented in Table 6.1, algorithms used the following values of XY precision: 0.2 mm, 0.1 mm, 0.05 mm, 0.02 mm and 0.01 mm. These values were chosen to reflect the most common resolutions

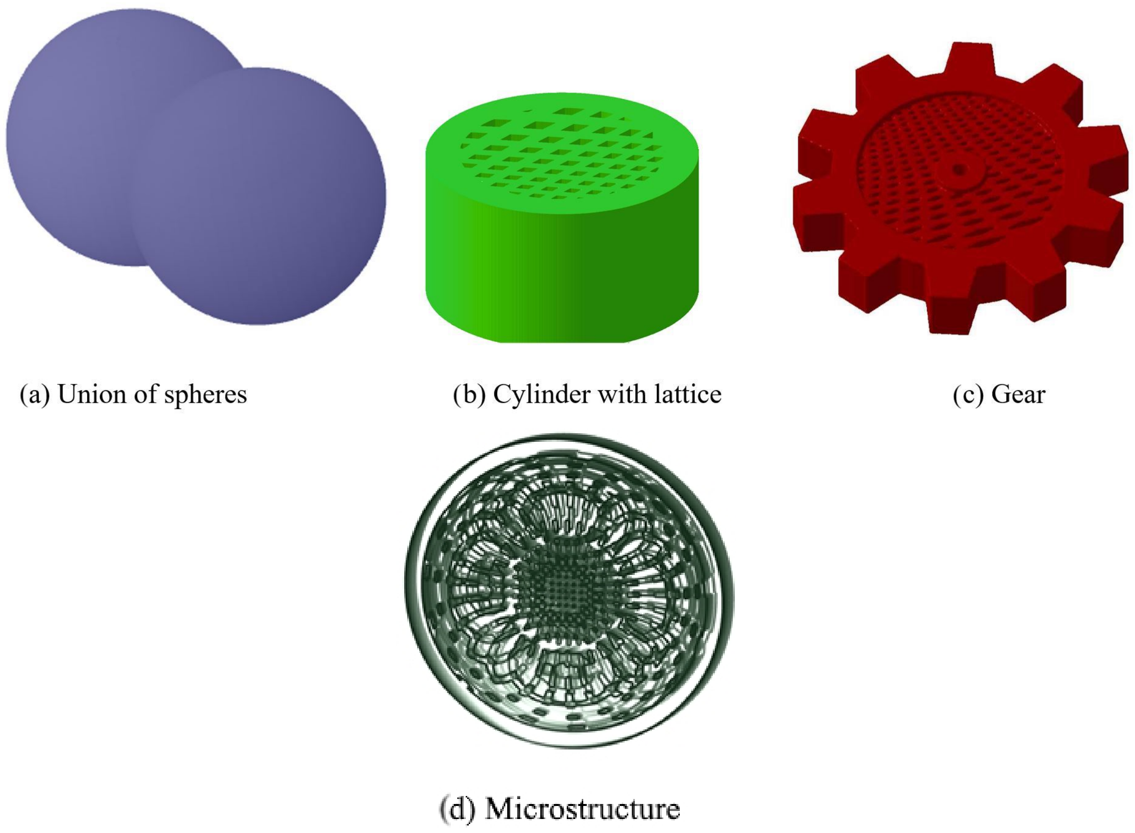


Figure 6-9: Non-algebraic surfaces.

of AM devices.

Model	Width, mm	Length, mm	Height, mm
Sphere	9	9	9
Decocube	4	4	4
Orthocircle	4	4	4
Union of two spheres	7	7	5
Cylinder with a lattice	60	60	60
Gear with a lattice	134	134	30
Microstructure	32	32	20

Table 6.1: Bounding boxes for contouring.

Figure 6-10 and Figure 6-11 allow us to see correlations in timings. For a more accurate comparison of the experimental results, statistics were used. F-tests for variance and t-tests for mean values were performed. All tests were performed with a p-value equal to 0.01 and a number of experiments equal to 20. Analysis of the plots and related statistics led us to the conclusions below.

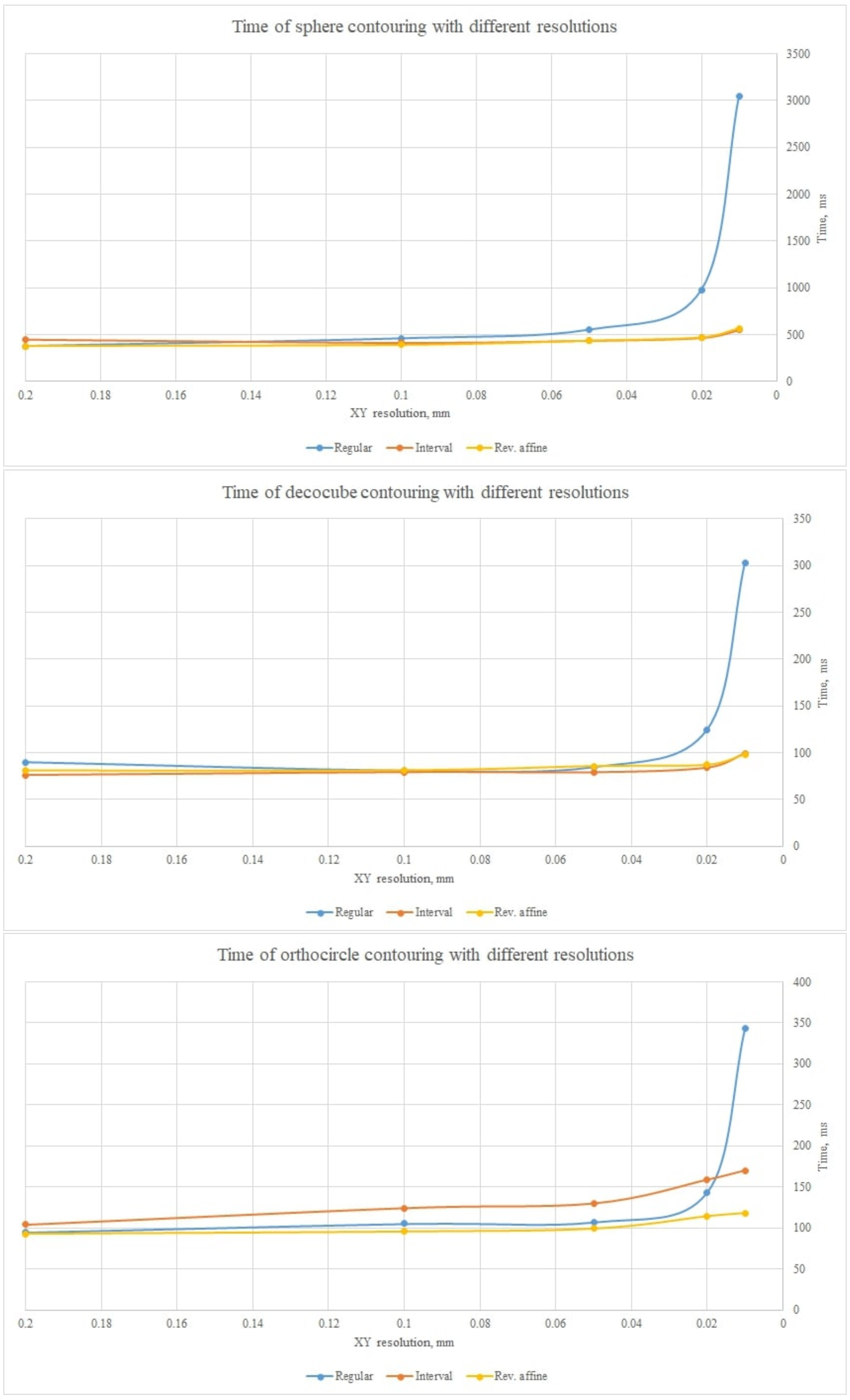
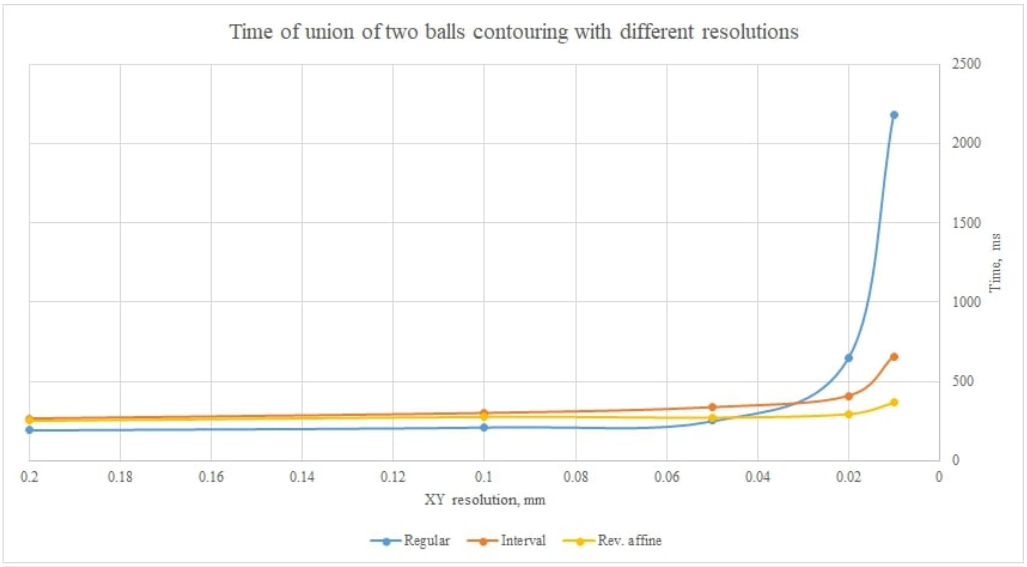
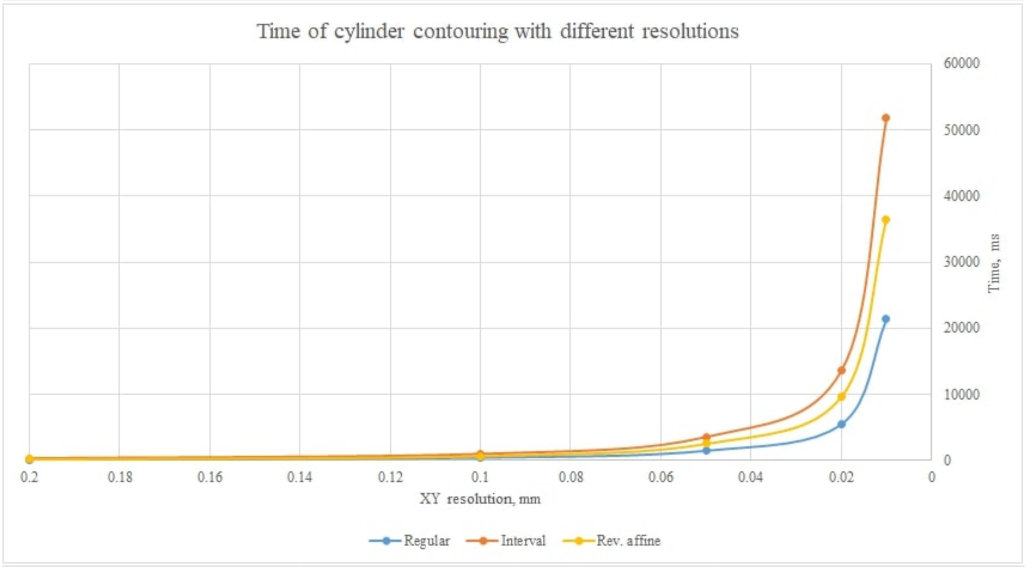
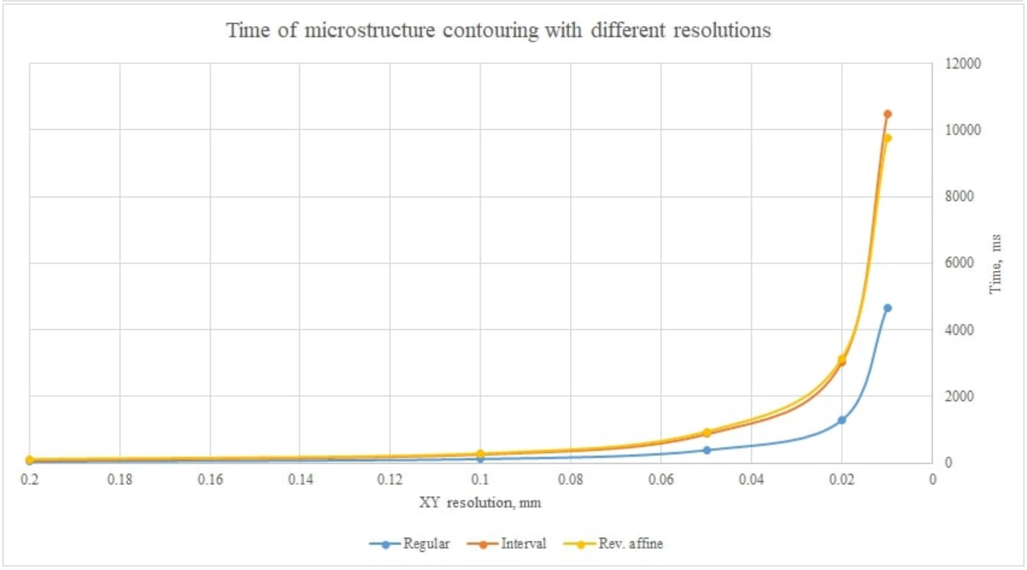


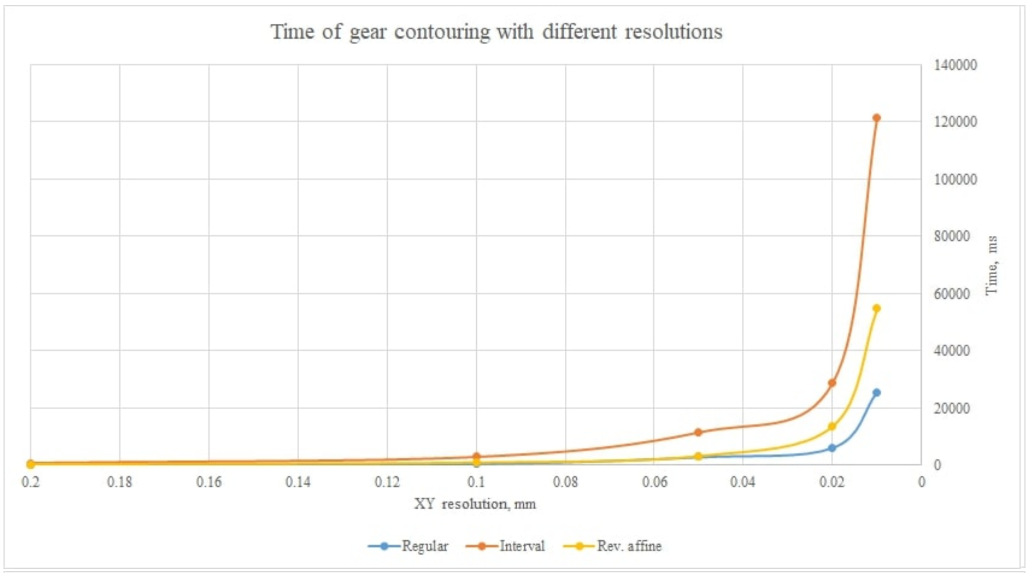
Figure 6-10: Contouring results for algebraic surfaces.

It can be seen that for algebraic surfaces with a relatively simple defining function, the adaptive techniques and conventional methods provide similar results on a rough resolution (see Figure 6-10). However, for a finer resolution, exhaustive


Time of union of two balls contouring with different resolutions
2500
2000
1500
1000
500
0
Time, ms
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
XY resolution, mm
Regular
Interval
Rev. affine


Time of cylinder contouring with different resolutions
60000
50000
40000
30000
20000
10000
0
Time, ms
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
XY resolution, mm
Regular
Interval
Rev. affine


Time of microstructure contouring with different resolutions
12000
10000
8000
6000
4000
2000
0
Time, ms
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
XY resolution, mm
Regular
Interval
Rev. affine



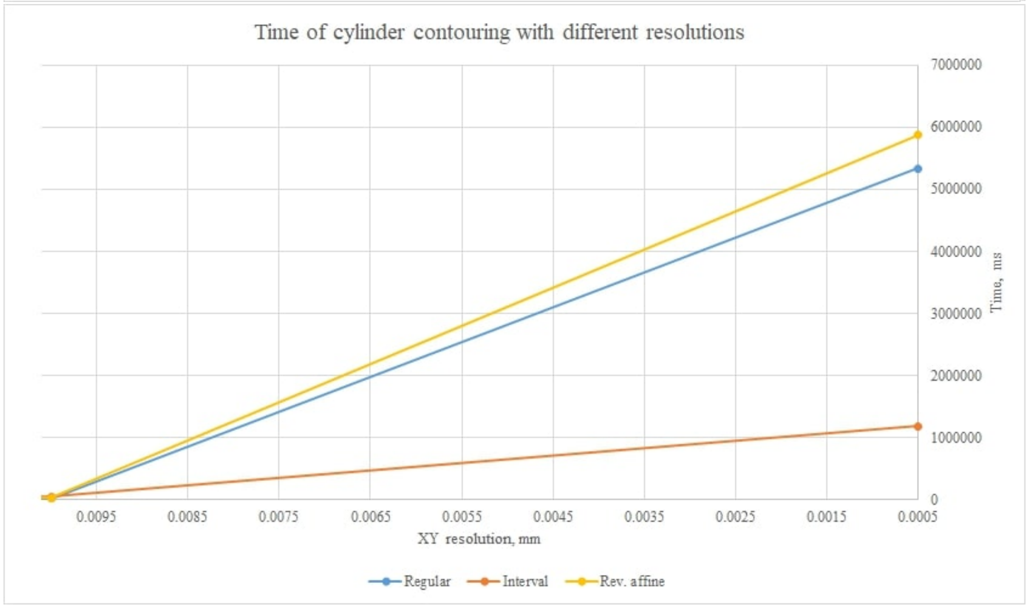


Figure 6-11: Contouring results for non-algebraic surfaces.

enumeration is no longer efficient.

For the sphere model and a resolution of 0.2 mm (200 μm), exhaustive enumeration and adaptive algorithms are the same with respect to time consumption. However, starting from a resolution of 100 μm, adaptive algorithms perform better. For the slightly complex algebraic model of the decocube, the exhaustive enumeration algorithm shows the same contouring time as adaptive algorithms until the resolution reaches 50 μm. For the orthocircle model, exhaustive enumeration works faster than the adaptive algorithm with IA for resolutions coarser than or equal to 20 μm. Its performance is relatively the same as that of the adaptive algorithm with revised AA up to a resolution of 50 μm and becomes worse for finer grids.

For the union of two balls, which is the simplest non-algebraic model considered, conventional contouring with exhaustive enumeration is faster when the resolution is above 0.05 mm. With grid sizes of 20 μm and 10 μm, adaptive methods work better. Note that only one non-algebraic operation is used in this model, which is the square-root operation. A more complex model of the cylinder with a lattice includes trigonometric functions in its definition in addition to the square-root operation. The exhaustive enumeration algorithm shows better results on this model for all considered resolutions. The same conclusions can be drawn for the microstructure and for the gear model (see implementations in Appendix B). For all considered resolutions, the algorithm with a regular grid works faster. The performance of both adaptive algorithms are the same.

One additional calculation with a 5 μm resolution was performed for the cylinder model (see Figure 6-11, bottom plot). It showed that the adaptive algorithm with IA is the best option in these conditions. It should be noted that such computations have a large cost in terms of both time and memory.

Thus, the difference in execution time between algorithms based on adaptive criteria, such as IA and AA, and the exhaustive enumeration algorithm increases with increasing precision of the calculated curve in one layer. Additionally, IA and AA work better with algebraic surfaces, especially with quadratic ones. However, the error of overestimation increases when transcendental functions or loops are used for describing complex models. The time efficiency of adaptive contouring techniques is more notable for computations with high XY resolution.

Our recommendations for using algorithms for contouring functionally defined 3D objects are summarized in Table 6.2. Note that the selected threshold (10 μm, 50 μm) changes depending on the model complexity and its original size. It moves to a more accurate XY resolution (smaller step size) with a more complex or smaller model. That means that with increasing model complexity, the use of adaptive methods becomes reasonable only with better accuracy. It relates to the number of operations used in the model definition and their computational complexity. The computation of adaptive criteria for these models takes more time than the time benefit from empty regions passing. Moreover, increasing the number of non-affine computational operations leads to interval overestimations (see [Fryazinov et al., 2010]). A finer computational grid provides a larger number of empty regions. The adaptive subdivision with revised AA is preferred concerning the choice between adaption criteria.

Implicit 3D objects	**XY resolution**	**Recommended contouring methods**
Algebraic surface without loops or conditions	$> 50\ \mu$m	exhaustive enumeration or adaptive algorithms
	$\leq 50\ \mu$m AND $> 20\ \mu$m	adaptive algorithms are more preferable
	$\leq 20\ \mu$m	only adaptive algorithms
Non-algebraic complex surfaces	$> 10\ \mu$m	exhaustive enumeration
	$\leq 10\ \mu$m	adaptive algorithms

Table 6.2: Recommended contouring methods for implicit 3D objects.

Our results and recommendations are applicable to Fused Deposition Modelling (FDM) and Direct Metal Deposition (DMD) printing of complex models. Both methods (exhaustive enumeration and adaptive algorithms) can be used until the precision reaches 60 μm. Adaptive algorithms are preferable for selective laser sintering and selective laser melting and stereolithography with DLP due to their high resolution near 10 μm.

6.2 Feature-based contouring

This section delivers the results mainly provided in [Maltsev et al., 2021]. It considers the contouring algorithm that can be used for FRep models of special form.

The paper focuses on FRep models with complex geometry as microstructures built via the replication of unit cells. The defining function of such FRep models can be described by the following statement:

$$f(x_1, x_2, \ldots, x_n, g(x_1, x_2, \ldots, x_n)). \tag{6.8}$$

In this equation, $g(x_1, x_2, \ldots, x_n)$ defines a feature of the initial model that can be represented by:

$$g(x_1, x_2, \ldots, x_n) = \gamma[u(x_1, x_2, \ldots, x_n)].$$

where $u(x_1, x_2, \ldots, x_n)$ defines a basis object, which is replicated over the modeling domain and is generally transformed by the operation γ. Further, $u(x_1, x_2, \ldots, x_n)$ will be named as a unit cell function or just a unit cell. For the operator γ a function Γ with the following properties should exist. The function Γ defines a sequence $B_i = (X_1, X_2, \ldots, X_n)_i, i = \overline{1, m}$ of bounding boxes for all replicated and transformed $u(x_1, x_2, \ldots, x_n)$ inside the considered domain. $X_1, X_2, \ldots, X_n$ are intervals along coordinate axes, which define the bounding boxes. The bounding boxes must not overlap, hence: $B_i \bigcap B_j = \varnothing$ for any $i, j = \overline{1, m}, i \neq j$. The consideration of unit cells with L-shape is beyond the scope.

The thesis proposes an accelerated contouring algorithm for such FRep models satisfying Equation 6.8. The algorithm applies a new compound adaptive criterion and a new acceleration criterion.

The criteria are based on binary search trees (B-trees), k-dimensional (k-d) trees and R-trees. The result structures, called spatial indexes, allow for the significant acceleration of the contouring process.

6.2.1 Search trees

B-trees assign a spatial index to each part of the 3D model. The indexes can be adapted to the scene geometry. It allowing the proper arrangement of 3D model elements and enabling fast access to, insertion of or removal of particular elements to/from the B-tree. Spatial indexes such as k-d trees, R-trees or bounding volume hierarchies are broadly used in 3D graphics, in particular, in ray-tracing algorithms [Havran and Bittner, 2002]. In [Foley and Sugerman, 2005], a k-d tree is used as an acceleration structure of the ray-tracing algorithm on a GPU. In [Wen et al., 2006], a k-d tree is used for surface reconstruction from a point cloud. R-trees are also used to accelerate the ray-tracing algorithm in [Feldmann, 2015] with results comparable to using k-d trees. Some tree-based representations have been used to allow efficient computation of multiscale vector volumes based on signed distance functions [Wang et al., 2011]. Trees in the form of bounding volume hierarchies have been previously used to accelerate the direct rendering of meta-balls [Gourmel et al., 2010].

A k-d tree is a space-partitioning data structure used for organizing data in k-dimensional space. They offer spatial querying methods such as a nearest neighbour search or a range search [Bentley, 1975]. A typical k-d tree is constructed hierarchically via division of the space with axis-aligned hyperplanes. Such a division allows for the reduction of the search range in the space. The performance of k-d trees depends on the strategy selected for space partitioning during k-d tree construction. Several strategies help in choosing a divider hyperplane to split the space: 1) the largest side of the bounding box 2) the median (the middle element of a sorted array) 3) the middle of each coordinate side of the bounding box, and 4) the surface area heuristic. The last one is a strategy developed for finding the best splitting plane for the optimal k-d tree construction [MacDonald and Booth, 1990]. A typical dataset for using a k-d tree is a point cloud, but the extension of a structure of a k-d tree allows for processing a set of k-dimensional rectangles (representing the bounding boxes of scene objects) [Havran et al., 1998].

An R-tree is a height-balanced tree [Guttman, 1984] similar to a binary search tree. This type of trees handles minimal Axis-Aligned Bounding Boxes (AABB) in n-dimensional space. The R-trees support spatial queries such as obtaining the nearest neighbourhood, checking for the intersection/containment of bounding boxes, inserting elements, and removing elements. An R-tree provides the equal depth for each leaf node and maintains a certain number of elements between the minimum and maximum at each node (excluding the root node) of the tree. Each leaf node contains an AABB of a single object. The construction of an R-tree is usually performed with the help of the insert operation. The tree construction strategy is based on maintaining the height of the tree branches and minimizing the overlap between the bounding boxes and the empty space. It includes a choice between a subtree insertion point and the division of overflowing nodes. These two criteria influence the performance of an R-tree. They were improved in the R+-tree for avoiding overlapping bounding rectangles [Sellis, 1987], and in the R*-tree, where the overlapping of nodal regions of the tree was minimized, as shown in [Beckmann et al., 1990] and [Beckmann and Seeger, 2009].

K-d trees and R-trees are similar data structures, they both split nodes with respect to locations in space determined by some heuristic. But the R-trees partition scenes with respect to the objects using their minimal AABBs, whereas k-d trees partition the underlying space.

However, the resulting partitions in the R-tree are not necessarily strictly binary partitions of the underlying space, as is the case with k-d trees. In addition, unlike the k-d tree, the R-tree is height-balanced and can be constructed incrementally by sequentially inserting data.

6.2.2 Feature-based contouring algorithm

The proposed feature-based contouring algorithm for the FRep models consists of the quadtree construction and the marching squares algorithm with the adaptive criteria. The marching squares algorithm is used to construct the contour topology, quadtrees and adaptive criteria are used to speed up calculations.

The quadtree is built from the root node, which is the bounding box of the implicit curve. Then, it is divided into four equal regions, which form the child nodes. Every child node can be recursively divided further. The process of contour extraction using the quadtree for one Z-layer is shown in Figure 6-12(a). The contouring process requires to calculate the value of the defining function of FRep model at very large number of points. An adaptive subdivision is used for decreasing the number of calculations of the defining function by localize the areas (cells of quadtree) which contain the implicit curve. Adaptive subdivision can employ an interval [Moore, 1966],[Sunaga, 2009] or affine arithmetic [Stolfi and de Figueiredo, 2003] as common adaptive criteria [Stolte and Kaufman, 1998], which allows you to evaluate the upper and lower bounds of the range of values of the defining function for each cell of the quadtree and decide whether to split the cell or not. In interval arithmetic, every quantity is represented by an interval of real numbers. An interval $[a, b]$ is a set of $x \in \mathbb{R}$ such that $a \leq x \leq b$. During functions processing in interval arithmetic, each quantity is replaced by its interval extension, and all computations are executed on intervals. If the upper and lower bounds of the interval are on opposite sides of zero, this means that the processed quadtree cell contains contours of the curve and should be divided.

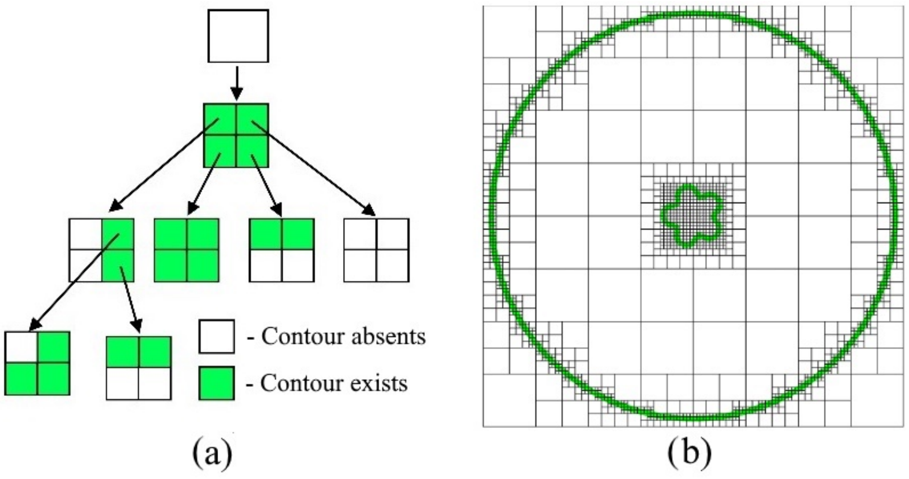


Figure 6-12: The contour extraction process for one layer of the FRep model: (a) building a quadtree, (b) contouring of one layer using the quadtree.

In most cases, FRep models is constructed using the loop operation. Their contouring is a computationally complex task. There are two main problems with 2D contour extraction from FRep models equipped with the loop operation. The first problem is that common adaptive criteria (interval, affine or revised affine arithmetic) do not work properly with such models due to interval overestimations (see Figure 6-13(a)). The intervals of values of the defining function becomes wide, and the algorithm divides all the cells of the quadtree without adaptivity. The second problem is the slow performance of the loop operation.

The proposed solution to both problems is to use an acceleration method based on the spatial indexes. The thesis presents the accelerated contouring algorithm, which uses this technique.

The accelerated slicing algorithm for FRep models satisfying Equation 6.8 contains the following steps:

An acceleration of the proposed slicing algorithm is reached by applying the contouring algorithm, which consists of the following steps:

Thus, the proposed feature based contouring algorithm is similar to Algorithm 7. However its `BuildQuadTree` procedure uses condition `CompoundAdaptiveCriteria`

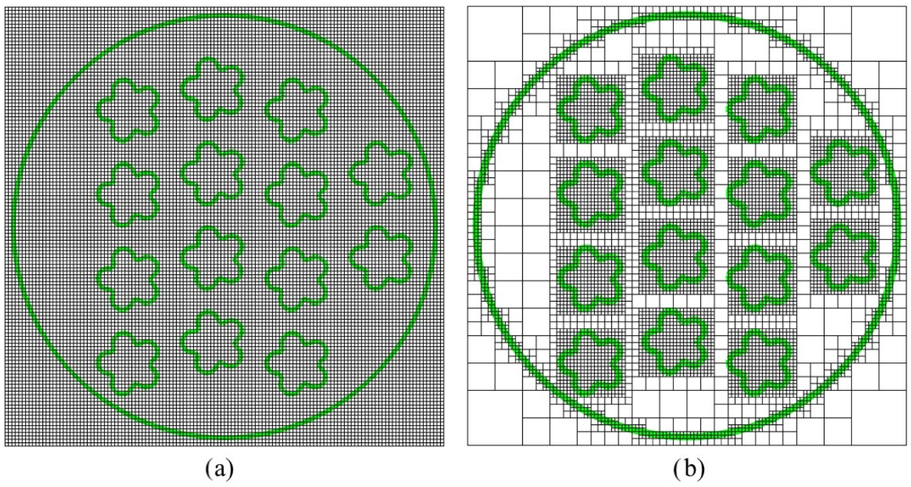


Figure 6-13: The contour extraction of one layer of the FRep model (contours - green lines, quadtree cells – black lines): (a) using a common adaptive criterion (b) using the compound adaptive criterion.

Algorithm 8: The accelerated slicing algorithm

```
Step 1. Cut the 3D model into layers with a certain step;
Step 2. Layer contouring;
Step 3. Post-processing the obtained contours with support and hatching;
Step 4. Generate CNC program for a specific 3D printer;
```

$(x, y, z, CellLength)$ `== TRUE` instead of `AdaptationCriterion` $(x, y, z, CellLength)$ `contains 0`. `CompoundAdaptiveCriteria` is listed below. `Intersections` function mentioned in this algorithm performs spatial search of point intersections with bounding boxes of unit cells.

The detailed description of the compound adaptive criterion and the acceleration criterion is given in Subsection 6.2.3 and Subsection 6.2.4. The connected component labelling algorithm was implemented according to [Shapiro, 1996]. The simple bisection method is used for solving the nonlinear equation (the defining function of the FRep model in the contouring process).

The process of constructing the spatial index (k-d tree) with 2D rectangles is shown in Figure 6-14(b) and (c). The median threshold strategy is used for the space division by a splitting plane. Every construction step divides the space at the median point of the arranged bounding boxes by X- and Y-aligned planes alternately. Unit cells on the left side of the median point are stored in the left subtree, and those on the right side of the median point are stored in the right subtree. The

Algorithm 9: The feature-based contouring algorithm

```
Step 1. Construct 2D bounding boxes $B_i = (X_1, X_2, \ldots, X_n)_i, i = \overline{1, m}$ for
  all unit cells $u(x_1, x_2, \ldots, x_n)$ of the FRep model (see Figure 6-12(a) red
  rectangles);
Step 2. Build the spatial index structure with calculated bounding boxes of
  unit cells in each layer of the sliced 3D model (Figure 6-12);
Step 3. Apply the compound adaptive criterion based on the spatial query
  during quadtree construction for the FRep model (see Figure 6-14(a));
Step 4. Applying the acceleration criterion based on the spatial search
  during calculating the defining function at every point of interest in the
  space;
Step 5. Create the topology of the curve using the marching squares
  algorithm and the connected component labelling algorithm [Shapiro,
  1996];
Step 6. Calculating the exact values of the implicit curve on the edges of
  adjacent cells using numerical methods for solving nonlinear equations;
```

Function CompoundAdaptiveCriteria($x, y, z, CellLength$)

```
if $AdaptationCriterion(x, y, z, CellLength)$ contains 0 then
    return $TRUE$;
else
    if $Intersections(x, y, z, CellLength) > 0$ then
        return $TRUE$;
return $FALSE$;
```

process stops after all the unit cells have been arranged or the width of the box of the leaf node is minimized. Every leaf of the k-d tree stores only one unit cell.

Instead of a k-d tree, an R-tree can also be used as the spatial index in Figure 6-14(d) and (e). There are linear and quadratic R-tree creation algorithms. The quadratic algorithm splits an overflow node by the two nodes with a minimal area of bounding boxes. The linear algorithm uses a splitting strategy based on the maximal distance between bounding boxes. The proposed algorithm used the quadric R-tree creation algorithm, where the maximal number of elements in the node equals 8 and minimal number of elements in the node equals 2. This algorithm was chosen because of the simplicity of its implementation and the availability of non-commercial libraries with this algorithm. Also, hierarchical structures such as bounding volume hierarchies (BVHs) [Gourmel et al., 2010] can be used as the spatial index.

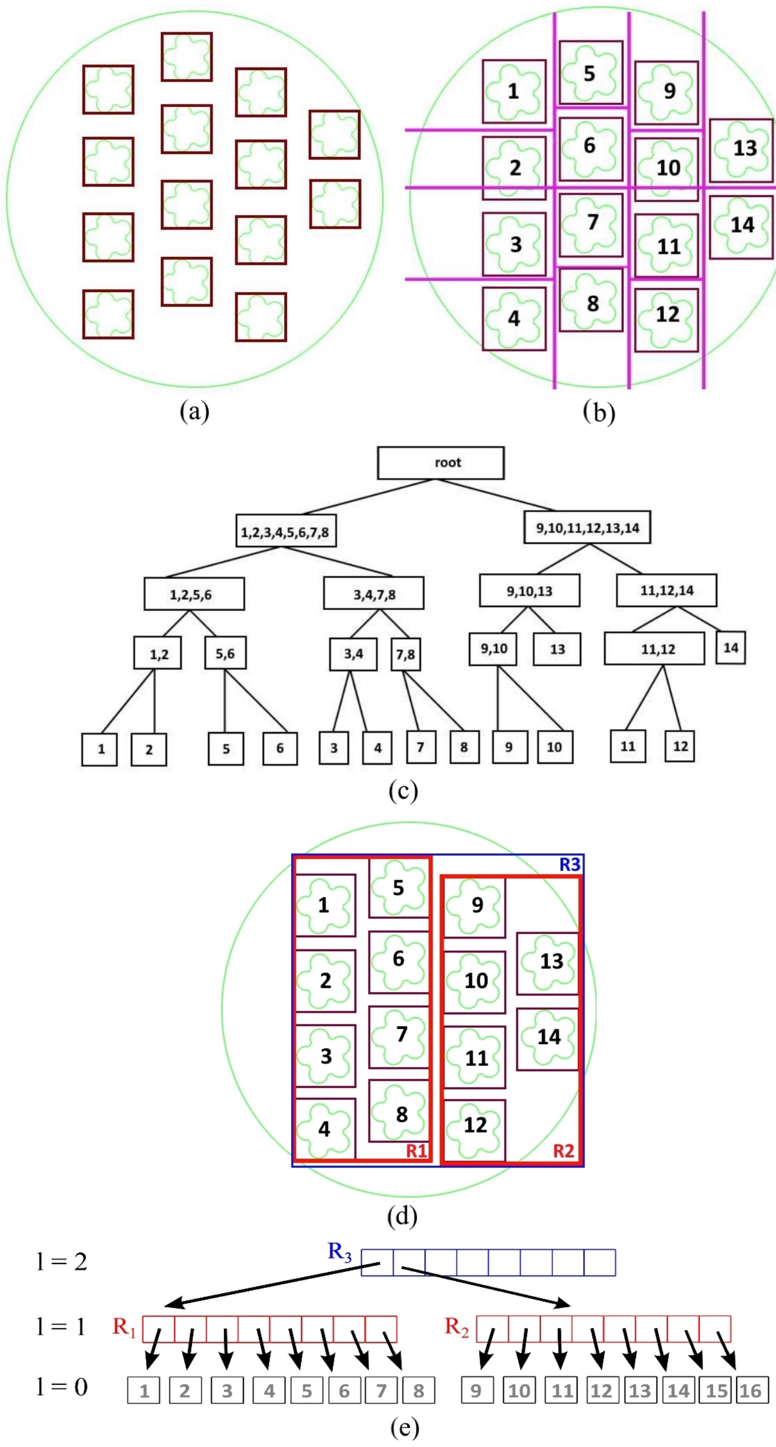


Figure 6-14: The process of k-d tree and R-tree construction for one layer of the FRep model: (a) bounding boxes of the unit cells (green lines – extracted contours, red rectangles – bounding boxes), (b) k-d tree construction, division of the space (purple lines), (c) kd-tree, (d) R-tree construction (max. number of elements in the node = 8 and min. number of elements in the node = 2), (e) R-tree.

6.2.3 The proposed compound adaptive criterion

Application of the common adaptive criteria for contouring of FRep models with the loop operations faces the problem of the overestimation of the intervals [Fryazinov et al., 2010]. It leads to the processing of almost all the quadtree cells in the space. In this case the adaptive contouring approach becomes the exhaustive enumeration method (see Figure 6-13(a)). In the accelerated contouring algorithm during the quadtree construction process, the decision of dividing each cell of the quadtree is made based on the proposed compound adaptive criterion.

Let us divide the initial FRep model by the boundary part and the microstructure part consisted of unit cell repetition. The proposed compound adaptive criterion consists of the common adaptive criterion for the boundary of the 3D model (interval or affine arithmetic) and the spatial queries, which are applied to the microstructure. It works as an adaptive criterion for each quadtree cell in the following way: if the interrogated quadtree cell contains at least one bounding box of the microstructure unit cell, this criterion returns true. The spatial query (intersection of the bounding boxes) is used for verification. Thus, the proposed compound adaptive criterion obtains its value from the common adaptive criteria for the boundary of the 3D model and from the intersection operation with the spatial search. If at least one of these two conditions is true, the quadtree cell is divided for further computations.

The proposed compound adaptive criterion optimizes calculation of the FRep defining function. It reduces the number of processed cells as shown in Figure 6-13(b). More details about the efficiency of the proposed compound adaptive criterion are shown in Subsection 6.2.5.

6.2.4 The proposed acceleration criterion

Applying the spatial search solves the performance problem for the loop operation by reducing the number of iterations. It allows for the evaluation only of a part of an FRep model.

The distance field calculation at every point of interest in the space leads to the computation of all the loop operations to build all the parts of the microstructure.

It performs, even if these parts are very far from the point of interest and do not influence its distance field value (see Figure 6-15(a)). Applying the spatial index allows for processing of only the unit cell closest to the point of interest. The Euclidean distance to the bounding box is used to find the nearest unit cell. Thus, the loop operation is replaced by a spatial search of the unit cell closest to the contours construction. During the calculation of the defining function of the FRep model, only the nearest unit cell (channel number 1 in Figure 6-15(b)) is taken into account for the computation, replacing the loop operation.

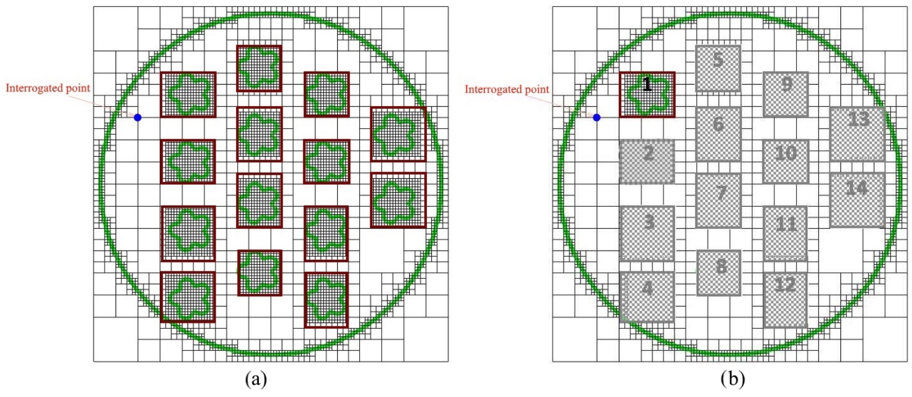


Figure 6-15: The process of contour extraction in one layer of the FRep model with the proposed compound adaptive and acceleration criteria: green lines – extracted contours, black lines – quadtree structure, red rectangles – precalculated bounded boxes of each channel of microstructure, blue point – point of interest.

The computational complexity of the search of the nearest elements to the point of interest is close to O $(\log n)$ using a spatial index, in contrast to performing the full loop operation with complexity O (n). The spatial index based on the k-d tree or R-tree is constructed for every layer of the sliced 3D model and is used to calculate the distance field value for 2D contour construction. It should be noted that the performance can be further improved by applying interval or affine arithmetic to the found unit cells.

Using this criterion makes the defining function of the FRep model discontinuous. It does not affect the contouring process, but the obtained model cannot be used as a primitive in further modeling process. This means that this criterion has to be applied at the end of the modeling process.

The accelerated contouring algorithm was implemented using C++17. The template-based Boost library was used for R-tree construction with the quadric creation algorithm Gehrels et al. [2009]. ALGLIB 3.16.0 was used to construct the k-d tree with the median threshold strategy for space division ALGLIB-Project [1999].

Two examples of the application of the accelerated contouring algorithm will be given in the following sections. The tests for these case studies were performed using the following hardware: a laptop with Intel Core i5-8250U CPU @ 1.80 GHz and 8 GB RAM.

6.2.5 Results of feature-based contouring

Case study 1: Contouring of a filter model

The first studied model is a filter for natural gas separation (see Figure 6-16). This model can be used in optimization tasks, where the new geometry of the filter can be obtained by changing its parameters (see Figure 6-16(c)). The reason for using the loop operation in the filter model is the complicated topology of its channels. Every channel should be placed in a specific area and rotated at different angles to reach the required density of the channels and the porosity of the whole filter. When the diameter of the channels is very tiny, the number of ones increases dramatically. This phenomenon leads to an increase in the contouring time. The unit cell in this model is one channel.

The filter model has 13 parameters: the filter height; filter radius; channel radius; N-pointed star channel; Maximal angle channel rotation; distance between channels; thickness shell; shift of the channel from the centre; maximal number of channels; type filling for the channels; shape of the channel entry hole; and the shape of the channel Z line. Different shapes of channels are shown in Figure 6-17.

The unit cell function $u(x, y, z)$ defines one channel of the filter FRep model. It is built from the profile function $\omega(x, y)$. The set of used profiles is shown in Figure 6-18. The simplest circle shape (see Figure 6-18(a)) is defined by

$$\omega(x, y) = R^2 - x^2 - y^2$$

The result $u(x, y, z)$ has the following form

$$u(x, y, z) = \omega(x', y'),$$

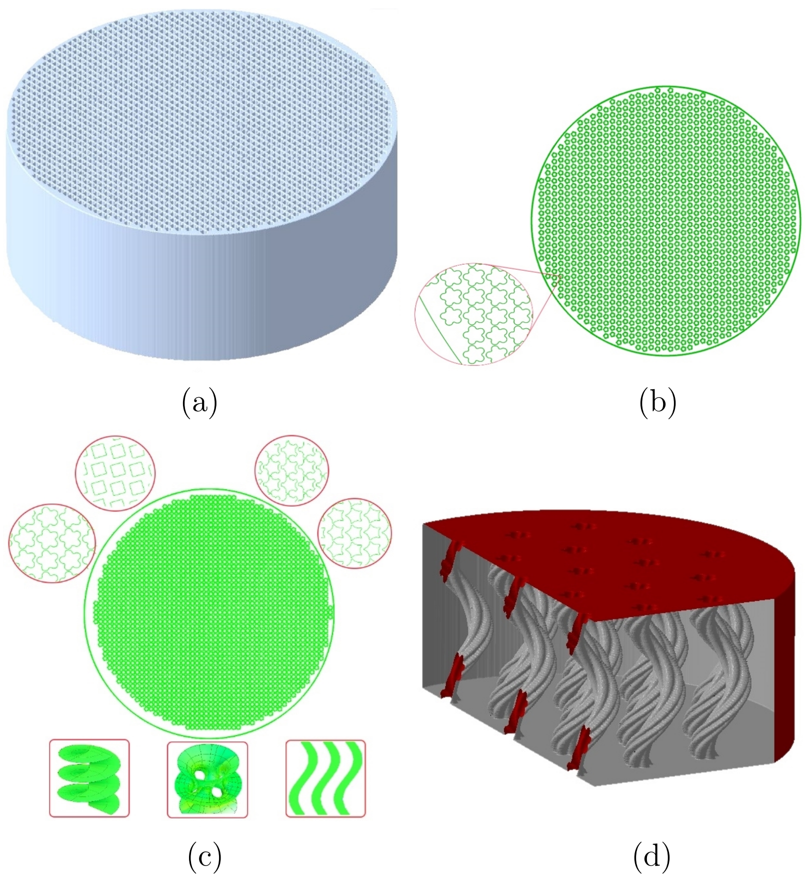


Figure 6-16: The filter model with channels: (a) the 3D object of the filter model; (b) cross-section of the 3D model with zoom; (c) Cross-sections of the filter model with parameterization examples; (d) the 3D object of the filter model with an X-Y cross-section.

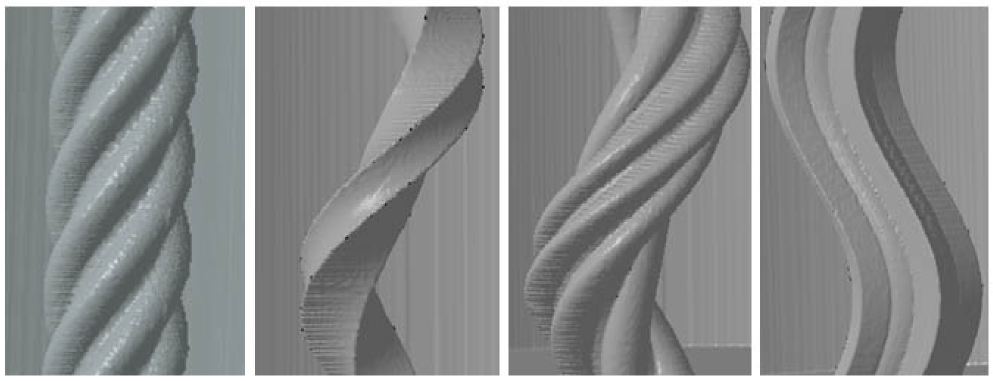


Figure 6-17: Different shapes of channels.

where the space mapping x', y' is defined by one of the following equations. For a sinusoidal mapping, it is

$$x' = x,$$

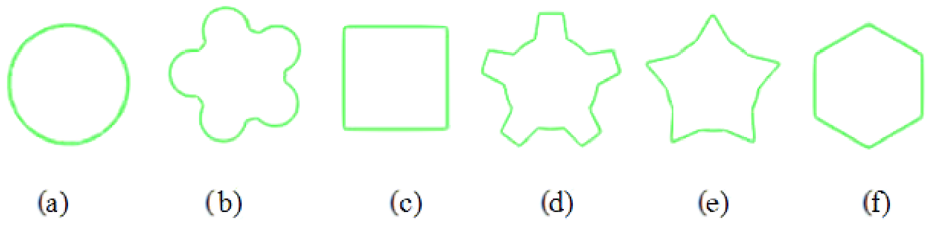


Figure 6-18: The shapes of the entry holes: (a) circle, (b) circle with petals, (c) square, (d) gear, (e) star, (f) hexagon.

$$y' = y + \sin(\theta) * \alpha.$$

For a twisting mapping, it is

$$x' = x * \cos(\theta) + y * \sin(\theta),$$
$$y' = -x * \sin(\theta) + y * \cos(\theta).$$

For a zigzag mapping, it is

$$x' = x,$$
$$y' = y + \tfrac{2*\arcsin(\sin(\theta))}{\pi} * \alpha.$$

In all the above-mentioned equations

$$t = \tfrac{z - z_1}{z_2 - z_1},$$
$$\theta = (1 - t) * \theta_1 + t * \theta_2,$$

where z_1 and z_2 are the bottom and top z coordinates of a channel, respectively; θ_1 and θ_2 are the rotation angles at the end points; α is the amplitude of the space mapping.

The operator of unit cell replication γ is

$$\gamma\left[u(x, y, z)\right] = \bigvee_{i=1}^{m} u(x - x_i, y - y_i, z)$$

where x_i and y_i are the centres of the channels in the loop operation and $\bigvee$ denotes the set-theoretic union.

The application of the feature-based contouring algorithm to the filter model with 2200 unit cells allows the reduction in the number of processed cells by 30% due to the proposed compound adaptive criterion. The contouring time was sped up 100-fold due to using the proposed compound adaptive criterion with the proposed acceleration criterion.

The tests were conducted 20 times for each filter model with different parameters (see Table 6.3) with an XY-resolution of 0.05 mm. The average values were calculated. The results of contouring the one layer of the filter model with a different number of channels and different shapes of the channel's entry holes are presented in Table 6.4, Table 6.5 and Figure 6-19. It should be noted that the shape of the channels (sinusoidal, twisting, zigzag) almost does not affect the contouring time.

Parameter	**Value**
Height of the filter	20 mm
Diameter of the filter	50 mm
Channel diameter	0.9-1.8 mm
Number of channels	500-2200

Table 6.3: The filter model parameters.

Number of unit cells (channels)	**Number of processed cells**		**Reducing % of the processed cells**
	Common adaptive criterion	**The proposed compound adaptive criterion**	
500	362479	244989	33
1000	362479	249269	32
2200	362479	250129	31

Table 6.4: The number of processed cells of one layer of the filter model.

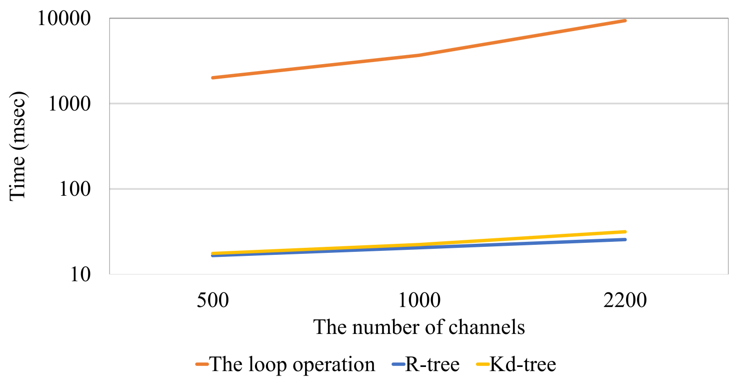


Figure 6-19: Plot of the experimental results with the filter model (the shape of the channel's entry holes: a circle with petals).

Table 6.5 shows that the use of spatial indexes significantly reduces the contouring time. At the same time, the R-tree and k-d tree data structures show almost

Channel diameter, mm	**Number of channels**	**Slicing time, msec**		
		Slicing with the full loop operation	**Accelerated slicing algorithm**	
			R-tree	**k-d tree**
circle				
1.8	500	63 552	1 273	1 222
1.5	1000	99 944	1 351	1 310
0.9	2200	297 326	1 683	1 779
circle with petals				
1.8	500	200 530	1 651	1 754
1.5	1000	367 219	2 042	2 230
0.9	2200	936 803	2 550	3 145
square				
1.8	500	82 073	1 432	1 875
1.5	1000	124 205	1 550	2 107
0.9	2200	390 451	1 970	2 584
gear				
1.8	500	1 063 341	2 794	3 328
1.5	1000	1 947 765	3 001	3 719
0.9	2200	4 454 432	4 214	5 205
star				
1.8	500	1 076 432	2 487	2 957
1.5	1000	1 971 207	2 703	3 436
0.9	2200	4 509 271	3738	4 411
hexagon				
1.8	500	129 222	1 436	1 906
1.5	1000	214 078	1 698	2 147
0.9	2200	634 755	2 008	2 591

Table 6.5: Experimental results of the filter model contouring step.

the same performance. The shape of the channel entry hole affects the contouring time due to additional calculations of the channel shape.

Case study 2: Contouring of a free-form model

This case consider contouring of 2D free-form object defined via Equation 5.3. For this FRep model the following function $\Gamma = \Gamma(x_i, y_i)$ exists:

$$\Gamma(x_i, y_i) = \{([x_i - h_x; x_i], [y_i; y_i + h_y]), ([x_i; x_i + h_x], [y_i; y_i + h_y]), ([x_i - h_x; x_i], [y_i - h_y; y_i]), ([x_i; x_i + h_x], [y_i - h_y; y_i])\}.$$

This function $\Gamma(x_i, y_i) : R \times R \to X \times X$, where $X \subseteq R$ is a set of intervals in R, returns the bounding boxes B_j which are quarters of the supports of the basis function $\phi_i(x, y)$.

The unit function for the bounding box B_j is:

$$u_j = \sum_k \alpha_k \phi_k(x, y)$$

where ϕ_k are basis functions with the supports that include B_j. There can be one to four ϕ_k functions in the sum.

Free-form functions are widespread in the task of topology optimization. The compliance minimization problem with boundary conditions shown in Figure 5-2(a) left was solved. The free-form function with plot presented in Figure 6-20(a) is a solution of this problem. It defines the solid body shown in Figure 6-20(b).

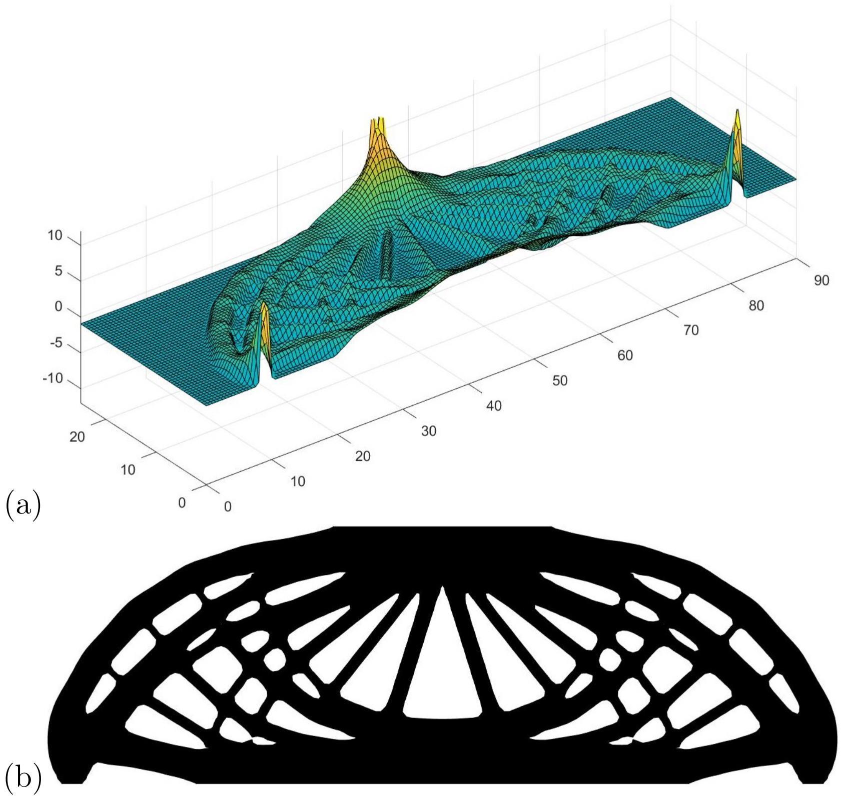


Figure 6-20: An example of a free-form model obtained via topology optimization algorithm: (a) the plot for the obtained free-form function $f(x, y)$ and (b) the optimized shape defined by the free-form function $f(x, y)$.

The proposed contouring algorithm is successfully applied to process the free-form FRep model. Its performance is 2-fold higher for the free-form model with 100x200 points if compared with the standard contouring algorithm (see Table 6.6 and Figure 6-21).

Number of points	Contouring time, msec		
	Slicing with all loop iterations	Accelerated slicing algorithm	
		R-tree	k-d tree
1250	1004	1000	960
5000	1699	1102	1143
20000	2800	1425	1300

Table 6.6: Experimental results for the free-form model.

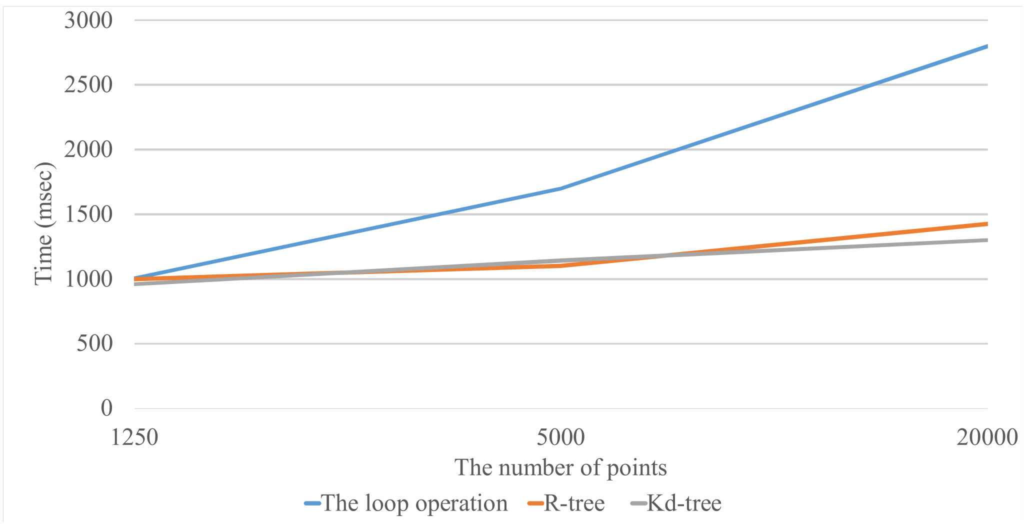
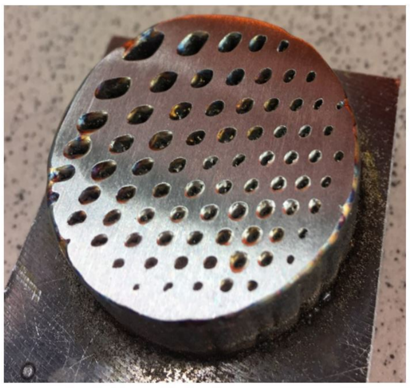
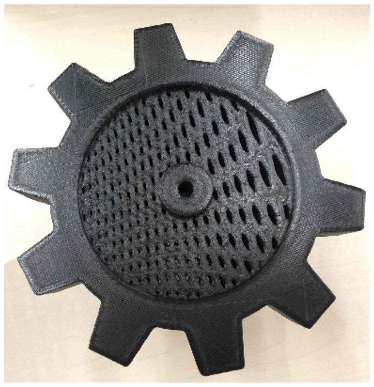


Figure 6-21: Plot of the experimental results for the free-form model.

6.3 3D printing of modeled parts

Test models were fabricated using FDM, DMD and DLP printing processes. The open-source project CuraEngine Ultimaker [2013] was used for the generation of supports, infill and GCODE for our FDM and DMD additive manufacturing equipment. In general, this software is applied for the FDM printing process; however, the strategies of FDM printing and DMD printing are different. Therefore, a specific AM profile was created for the 3D laser-aided Direct Metal Tooling (DMT) printer Insstek MX-1000. This profile includes the generation of specific GCODE commands for managing the feed speed of the metal powder, the gas shield and the laser beam. Additionally, DMT printing with the stainless steel metal powder requires printing the shells twice; only after that does the infill process start. The printed part using the 3D DMT printer Insstek MX-1000 is shown in Figure 6-22(a). The same software was used for the fabrication of the gear model using the 3D FDM printer Ultimaker S5 (see Figure 6-22(b)). The 3D DLP printer Wanhao Duplicator 7 was used for the fabrication of the microstructure model (see Figure 6-22(c)). Raster slices and GCODE for this model were prepared using our software module in CWS format.


(a) The cylinder with lattice


(b) The gear

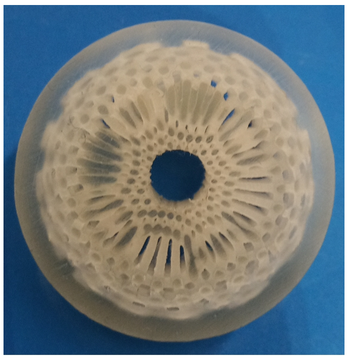

(c) The microstructure

Figure 6-22: Printed models: (a) the cylinder with lattice (radius 30 mm, height 60 mm) printed using the 3D DMT printer Insstek MX-1000; (b) the gear (radius 67 mm, height 30 mm) printed using the 3D FDM printer Ultimaker S5; (c) the microstructure (radius 16 mm, height 20 mm) printed using the 3D DLP printer Wanhao Duplicator 7.

Similarly, parts considered in Section 6.2 were manufactured. Ultimaker S5 FDM 3D printer was used to manufacture the filter model with radius equals $25mm$ and height equals 20mm (see Figure 6-23(a)). Wanhao Duplicator 7 DLP 3D printer was used to manufacture the free-form FRep model with length equals 100mm, width equals 50mm and height equals 20mm (see Figure 6-23(b)).

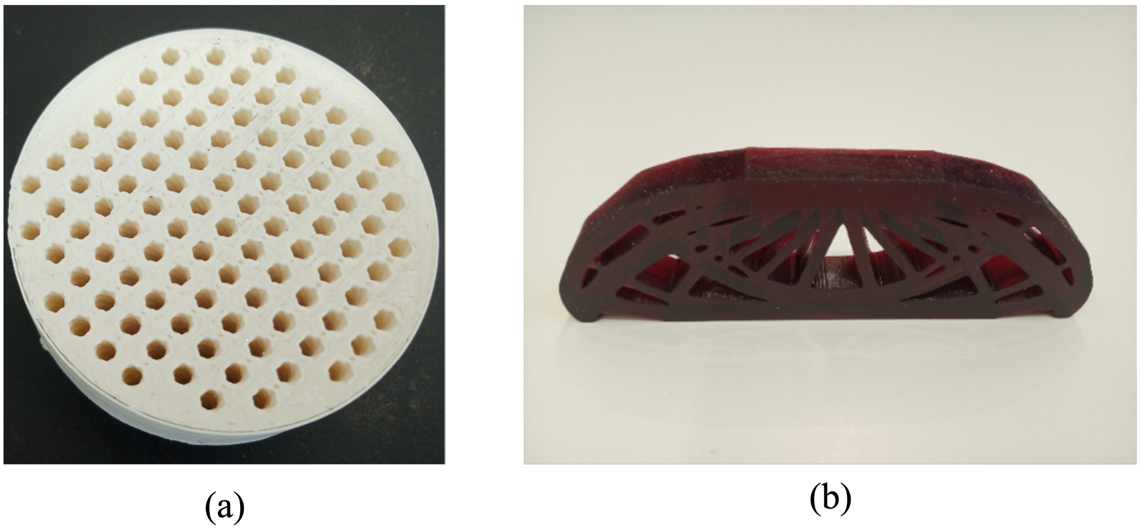


Figure 6-23: Printed models: (a) the filter model printed using an Ultimaker S5 3D FDM printer, (b) the free-form FRep model printed using a Wanhao Duplicator 7 DLP 3D printer.

"If you optimize everything, you will always be unhappy."

Donald Knuth

Chapter 7

Conclusion

This thesis studied opportunities of FRep in CAD/CAM and optimization for AM. The architecture and implementation of the CAD/CAM system based on FRep were proposed. The connection between FRep and the level set methods was established for structural optimization. Finally, experimental validation of the proposed system was performed.

Its main component is the FRep geometric core. It allows users to perform symbolic modeling of parameterized bodies. Fast rendering routines support the geometric core. They were implemented using parallel computations on GPU.

One more aspect of the proposed system is its compatibility with optimization algorithms. The thesis showed and discussed the connection between FRep and level-set optimization methods. The thesis proposes efficient and robust topology optimization algorithm and parameter optimization algorithm and their modifications. The thesis showed how one can capture shape constraints during the topology optimization and how to perform structural optimization that includes both topology and parameter optimization. The proposed algorithms can be extended to multimaterial objects.

The main part of the proposed CAM component is the efficient slicing algorithm. It appears that the classical approach based on marching squares is the most convenient choice for common cases of AM. It was compared with algorithms exploiting interval arithmetic and affine arithmetic, but these algorithms work significantly faster when the ratio of object size to slicing precision is greater than 10^5. For example, this ratio holds for parts with 10 mm size and 3D printing accuracy about 1 μm. Nowadays, you can meet such a case only in cutting edge AM equipment.

More practical cases were considered in the paper devoted to FRep-based CAM development [Maltsev et al., 2021]. Slicing become a problem when the object has a significant number of repeating structures such as pores, channels or balks. This thesis showed that these objects have a specific form of FRep defining functions, and this form can be exploited for fast slicing. The proposed algorithm based on K-d trees and R-trees can be up to 100 faster for the considered objects.

FRep-based approach for design and manufacturing was applied to implants prototyping. Several structures those satisfy 3D printing requirements were designed. Their mechanical tests and simulations were performed. This research showed that FRep-based methodology can be successfully applied for AM.

Delivered results allow us to propose the integrated CAD/CAM system based on FRep. The system can interact with developed topology optimization algorithm and use its output as a new FRep object. Designed models with incorporated optimized parts can be directly manufactured with 3D printers. Mechanical tests of printed parts validated of the developed simulation algorithms with and this validation was successful.

Therefore, this dissertation describes the novel modeling system for advanced 3D printing. It is the first proposed CAD/CAM system with a unique geometry representation containing modeling, optimization, and CNC code generation for AM components. The system resolves several challenges of excising commercial CAD/CAM software. It excludes errors of model geometry that come from transformations between optimization and modeling modules. Moreover, it avoids the smoothing of optimized parts because the proposed level-set-based optimization algorithms control the smoothness of the resulting geometry by appropriate choice of basis functions and knot point. These algorithms can be used with shape constraints and parameter optimization for more precise control of critical regions of a solid. Finally, the web-based architecture of the software allows arranging user collaboration for modeling tasks. The system uses proposed algorithms with improved time efficiency.

However, the proposed system and its core technology have several limitations. This thesis studied slicing techniques for FRep objects and proposed efficient algorithms for the generation of CNC programs for AM, but unfortunately they are useless for some AM equipment. This happens because of their closed architecture and ".stl" file format as the only option for model upload. Open program interfaces provided by manufacturers of 3D printers can resolve this limitation.

Fortunately, most AM machines have these interfaces. They allow us to upload CNC programs generated by our system to 3D printers. However, the size of the programs for objects with complex structure can exceed file limits of the equipment. The complete program can be split into several parts to overcome this obstacle.

One more limitation comes from the proposed optimization algorithm. The optimization problem is not convex. Thus, any proposed algorithm can guarantee only a local optimum. If it is crucial to obtain the best solution, then one can try to provide different initial geometries for the optimization algorithm or run it with several samples of optimization parameters, such as the time step or the number of relaxation steps. Despite these limitations the proposed approach for design, optimization and manufacturing seems promising for further researches.

One of the possible direction of future work is considering more optimization objectives, e.g. heat conduction and fatigue problems. New physical tasks are not the only source of new optimization tasks. Future works can be addressed to technological constraints of AM and their influence to the optimization algorithm. Therefore, works on structural optimization have an almost unlimited source of research problems.

All completed and proposed future studies are fascinating. However, their attractiveness is commensurate with their complexity, and one need to adequately assess their demand.

Bibliography

Kalayu Mekonen Abate, Aamer Nazir, and Jeng-Ywan Jeng. Design, optimization, and selective laser melting of vin tiles cellular structure-based hip implant. *The International Journal of Advanced Manufacturing Technology*, 112(7):2037–2050, 2021.

Osama Abdulhameed, Abdulrahman Al-Ahmari, Wadea Ameen, and Syed Hammad Mian. Additive manufacturing: Challenges, trends, and applications. *Advances in Mechanical Engineering*, 11(2):1687814018822880, 2019.

Abdulsalam Abdulaziz Al-Tamimi, Henrique Almeida, and Paulo Bartolo. Structural optimisation for medical implants through additive manufacturing. *Progress in Additive Manufacturing*, 5(2):95–110, 2020.

ALGLIB-Project. Data processing library, 1999. URL https://www.alglib.net/. last accessed 5 March 2020.

Alibre, LLC. Alibre Design, 2022. URL https://www.alibre.com.

Grégoire Allaire, François Jouve, and Anca-Maria Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, 194(1):363–393, 2004.

Sadettin Cem Altıparmak and Bowen Xiao. A market assessment of additive manufacturing potential for the aerospace industry. *Journal of Manufacturing Processes*, 68:728–738, 2021.

Eilam Amir and Oded Amir. Concurrent high-resolution topology optimization of structures and their supports for additive manufacturing. *Structural and Multidisciplinary Optimization*, 63(6):2589–2612, 2021.

Samuel Amstutz and Heiko Andrä. A new algorithm for topology optimization using a level-set method. *Journal of computational physics*, 216(2):573–588, 2006.

ANSYS, Inc. SpaceClaim, 2022. URL https://www.ansys.com/products/3d-design/ansys-spaceclaim.

Ariya Hidayat. Esprima, 2022. URL https://esprima.org/.

Army Research Laboratory. BRL-CAD, 2022. URL https://brlcad.org.

M. Attene, M. Livesu, S. Lefebvre, T. Funkhouser, S. Rusinkiewicz, S. Ellero, J. Martínez, and A.H. Bermano. Design, representations, and processing for additive manufacturing. *Synthesis Lectures on Visual Computing*, 10(2):1–146, 2018. doi:https://doi.org/10.2200/S00847ED1V01Y201804VCP031.

Autodesk Inc. AutoCAD, 2022a. URL https://www.autodesk.com/products/autocad/.

Autodesk Inc. AutoCAD Architecture, 2022b. URL https://www.autodesk.com/products/autocad/included-toolsets/autocad-architecture.

Autodesk Inc. Inventor, 2022c. URL https://www.autodesk.com/products/inventor/.

Autodesk Inc. Revit, 2022d. URL https://www.autodesk.com/products/revit/.

Autodesk Inc. Fusion 360, 2022e. URL https://www.autodesk.com/products/fusion-360/.

N. Beckmann and B. Seeger. A revised r*-tree in comparison with related index structures. In *SIGMOD Conference*, pages 799–812, 2009.

N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, SIGMOD '90, page 322–331, New York, NY, USA, 1990. Association for Computing Machinery. doi:10.1145/93597.98741.

Martin Philip Bendsoe and Ole Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2003.

J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. URL www.scopus.com. last accessed 5 March 2020.

Bentley Systems Inc. MicroStation, 2022. URL https://www.bentley.com/en/products/brands/microstation.

Pierre Bézier. *The mathematical basis of the UNIURF CAD system*. Butterworth-Heinemann, 2014.

Shajay Bhooshan, Johannes Ladinig, Tom Van Mele, and Philippe Block. Function representation for robotic 3d printed concrete. In *Robotic fabrication in architecture, art and design*, pages 98–109. Springer, 2018.

Ganesh H Bhosale and Sagar U Sapkal. Re-engineering of cast product by topology optimization. *International Journal*, 9(9), 2021.

J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988. ISSN 0167-8396. doi:https://doi.org/10.1016/0167-8396(88)90013-1. URL http://www.sciencedirect.com/science/article/pii/0167839688900131.

Jules Bloomenthal and Brian Wyvill. Introduction to implicit surfaces. 1997.

Bricsys NV. BricsCAD, 2022. URL `https://www.bricsys.com/en-eu`.

H. Brönnimann, G. Melquiond, and S. Pion. The design of the boost interval arithmetic library. *Theoretical Computer Science*, 351(1):111–118, 2006. ISSN 0304-3975. doi:https://doi.org/10.1016/j.tcs.2005.09.062. URL `http://www.sciencedirect.com/science/article/pii/S0304397505006110`.

Martin Burger, Benjamin Hackl, and Wolfgang Ring. Incorporating topological derivatives into level set methods. *Journal of computational physics*, 194(1):344–362, 2004.

Katja Bühler. Implicit linear interval estimations. In *Proceedings of the 18th Spring Conference on Computer Graphics*, SCCG '02, pages 123–132, New York, NY, USA, 2002. ACM. ISBN 1-58113-608-0. doi:10.1145/584458.584479.

cadwork Software. cadwork, 2022. URL `https://en.cadwork.com/`.

Hamish A. Carr, Thomas Theußl, and Torsten Möller. Isosurfaces on optimal regular samples. In *VisSym*, 2003.

Vivien J Challis and James K Guest. Level set topology optimization of fluids in stokes flow. *International journal for numerical methods in engineering*, 79(10): 1284–1308, 2009.

Yong Chen and Charlie CL Wang. Uniform offsetting of polygonal model based on layered depth-normal images. *Computer-aided design*, 43(1):31–46, 2011.

Corel Corporation. CorelCAD, 2022. URL `https://www.coreldraw.com/en/product/corel-cad/`.

Quentin Corker-Marin, Alexander Pasko, and Valery Adzhiev. 4d cubism: Modeling, animation, and fabrication of artistic shapes. *IEEE computer graphics and applications*, 38(3):131–139, 2018.

Dassault Systèmes. CATIA, 2022a. URL `https://www.3ds.com/products-services/catia/`.

Dassault Systèmes. DraftSight, 2022b. URL `https://www.draftsight.com`.

Dassault Systèmes. SOLIDWORKS, 2022c. URL `https://www.3ds.com/products-services/solidworks/`.

B. R. de Araújo, Daniel S. Lopes, Pauline Jepp, Joaquim A. Jorge, and Brian Wyvill. A survey on implicit surface polygonization. *ACM Comput. Surv.*, 47(4): 60:1–60:39, May 2015. ISSN 0360-0300. doi:10.1145/2732197.

L.H. de Figueiredo and J. Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37:147–158, 2004. ISSN 1572-9265. doi:10.1023/B:NUMA.0000049462.70970.b6.

Gouri Dhatt, Emmanuel Lefrançois, and Gilbert Touzot. *Finite element method.* John Wiley & Sons, 2012.

Leonardo S Duarte, Waldemar Celes, Anderson Pereira, Ivan F M Menezes, and Glaucio H Paulino. PolyTop++: an efficient alternative for serial and parallel topology optimization on CPUs & GPUs. *Structural and Multidisciplinary Optimization*, 52(5):845–859, 2015.

Tom Duff. Interval arithmetic recursive subdivision for implicit functions and constructive solid geometry. In *SIGGRAPH '92*, 1992.

ECMA. Introducing json, 2022. URL https://www.json.org/json-en.html. last accessed 27 September 2022.

ephtracy. MagicaCSG, 2021. URL https://ephtracy.github.io/index.html?page=magicacsg.

Stanley Osher Ronald Fedkiw and Stanley Osher. Level set methods and dynamic implicit surfaces. *Surfaces*, 44(77):685, 2002.

D. Feldmann. Accelerated ray tracing using r-trees. In *GRAPP 2015 - 10th International Conference on Computer Graphics Theory and Applications; VISIGRAPP, Proceedings*, pages 247–257, 2015.

T. Foley and J. Sugerman. Kd-tree acceleration structures for a gpu raytracer. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS '05, page 15–22, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1071866.1071869.

O. Fryazinov, A. Pasko, and P. Comninos. Fast reliable interrogation of procedurally defined implicit surfaces using extended revised affine arithmetic. *Comput. Graph.*, 34(6):708–718, December 2010. ISSN 0097-8493. doi:10.1016/j.cag.2010.07.003.

Oleg Fryazinov, Turlif Vilbrandt, and Alexander Pasko. Multi-scale space-variant frep cellular structures. *Computer-Aided Design*, 45(1):26–34, 2013.

B. Gehrels, B. Lalande, M. Loskot, and A. Wulkiewicz. Boost library. spatial indexes library, 2009. URL https://www.boost.org/doc/libs/1_60_0/libs/geometry/doc/html/geometry/reference/spatial_indexes.html. last accessed 5 March 2020.

Olivier Gourmel, Anthony Pajot, Mathias Paulin, LoÃ¯c Barthe, and Pierre Poulin. Fitted bvh for fast raytracing of metaballs. *Computer Graphics Forum*, 29, 05 2010. doi:10.1111/j.1467-8659.2009.01597.x.

Graphisoft. Archicad, 2022. URL https://graphisoft.com/ru/solutions/archicad.

Gstarsoft Co.,Ltd. progeCAD, 2022. URL https://www.gstarcad.net/cad/.

A. Guttman. R trees: A dynamic index structure for spatial searching. *Sigmod Record*, 14:47–57, 1984. doi:10.1145/971697.602266.

V. Havran and J. Bittner. On improving kd-trees for ray shooting. In *In Proc. of WSCG 2002 Conference*, pages 209–217, 2002.

Vlastimil Havran, Tomás Kopal, Jiri Bittner, and Jiri Zara. Fast robust bsp tree traversal algorithm for ray tracing. 12 1998. doi:10.1080/10867651.1997.10487481.

Brian Von Herzen and Alan H. Barr. Accurate triangulations of deformed, intersecting surfaces. In *SIGGRAPH '87*, 1987.

Pi-Chung Hsu. K-ary implicit blends with increasing or decreasing blend ranges for level blend surfaces. *Journal of Advances in Information Technology*, 2018.

HyperFun Team. HyperFun, 2022. URL `https://hyperfun.org`.

IMSI Design LLC. TurboCAD, 2022. URL `https://www.turbocad.com/`.

IntelliCAD Technology Consortium. IntelliCAD, 2022. URL `https://www.intellicad.org`.

IronCAD, LLC. IntelliCAD, 2022. URL `https://www.ironcad.com`.

Jytra Technology Solutions. TrueCAD, 2022. URL `https://truecad.com`.

D. Kalra and A. H. Barr. Guaranteed ray intersections with implicit surfaces. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, pages 297–306, New York, NY, USA, 1989. ACM. ISBN 0-89791-312-4. doi:10.1145/74333.74364.

Sandilya Kambampati, Carolina Jauregui, Ken Museth, and H Alicia Kim. Geometry design using function representation on a sparse hierarchical data structure. *Computer-Aided Design*, 133:102989, 2021.

VF Kravchenko, VE Antciperov, LE Nazarov, OV Kravchenko, DV Churikov, Ya Yu Konovalov, and KA Budunova. R-, atomic and wa-system functions theory in physical problems. In *2021 Photonics & Electromagnetics Research Symposium (PIERS)*, pages 2532–2541. IEEE, 2021.

Viktor F Kravchenko, Oleg V Kravchenko, Yaroslav Yu Konovalov, and Kristina A Budunova. Atomic functions theory: History and modern results: Dedicated to the pioneer of atomic functions theory vl rvachev invited paper. *2020 IEEE Ukrainian Microwave Week (UkrMW)*, pages 619–623, 2020.

Y.O. Kuzminova, D.G. Firsov, S.D. Konev, A.A. Dudin, S.A. Dagesyan, I.S. Akhatov, and S.A. Evlashin. Structure control of 316l stainless steel through an additive manufacturing. *Letters on Materials*, 9(4s):551–555, 2019. doi:10.22226/2410-3535-2019-4-551-555. URL `https://lettersonmaterials.com/en/Readers/Article.aspx?aid=23461`.

LeoCAD.org. LeoCAD, 2022. URL https://www.leocad.org.

Konstantin Levinski and Alexei Sourin. Interactive polygonisation for function-based shape modelling. 2002.

Hui Liu, Peng Wei, and Michael Yu Wang. Cpu parallel-based adaptive parameterized level set method for large-scale structural topology optimization. *Structural and Multidisciplinary Optimization*, 65(1):1–15, 2022a.

Jikai Liu, Andrew T Gaynor, Shikui Chen, Zhan Kang, Krishnan Suresh, Akihiro Takezawa, Lei Li, Junji Kato, Jinyuan Tang, Charlie CL Wang, et al. Current and future trends in topology optimization for additive manufacturing. *Structural and multidisciplinary optimization*, 57(6):2457–2483, 2018.

Junli Liu, Vuong Nguyen-Van, Biranchi Panda, Kate Fox, Anton du Plessis, and Phuong Tran. Additive manufacturing of sustainable construction materials and form-finding structures: a review on recent progresses. *3D Printing and Additive Manufacturing*, 9(1):12–34, 2022b.

W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

J.D. MacDonald and K.S. Booth. Heuristics for ray tracing using space subdivision. *The Visual Computer*, 6(3):153–166, 1990. ISSN 1432-2315. doi:10.1007/BF01911006.

Y Maksum, A Amirli, A Amangeldi, M Inkarbekov, Y Ding, A Romagnoli, S Rustamov, and B Akhmetov. Computational acceleration of topology optimization using parallel computing and machine learning methods–analysis of research trends. *Journal of Industrial Information Integration*, page 100352, 2022.

Evgenii Maltsev, Dmitry Popov, Svyatoslav Chugunov, Alexander Pasko, and Iskander Akhatov. Using function representation of 3d models in additive manufacturing. In *All-Russian Congress on the fundamental problems of theoretical and applied mechanics*, 2019.

Evgenii Maltsev, Dmitry Popov, Svyatoslav Chugunov, Alexander Pasko, and Iskander Akhatov. An accelerated slicing algorithm for frep models. *Applied Sciences*, 11(15):6767, 2021.

Marius Kintel. OpenSCAD, 2022. URL https://openscad.org/.

G. Melquiond, S. Pion, and H. Brönnimann. Boost library. interval arithmetic library, 2006. URL https://www.boost.org/doc/libs/1_66_0/libs/numeric/interval/doc/interval.htm. Last accessed 19 February 2019.

Liang Meng, Weihong Zhang, Dongliang Quan, Guanghui Shi, Lei Tang, Yuliang Hou, Piotr Breitkopf, Jihong Zhu, and Tong Gao. From topology optimization design to additive manufacturing: Today's success and tomorrow's roadmap. *Archives of Computational Methods in Engineering*, 27(3):805–830, 2020.

K. Mochizuki. Robust and adaptive polygonization of implicit curves and surfaces. Master's thesis, Aizu-Wakamatsu: Aizu University, 2004.

R.E. Moore. *Interval Analysis*. Englewood Cliff, New Jersey: Prentice-Hall., 1966.

Sougata Mukherjee, Dongcheng Lu, Balaji Raghavan, Piotr Breitkopf, Subhrajit Dutta, Manyu Xiao, and Weihong Zhang. Accelerating large-scale topology optimization: state-of-the-art and challenges. *Archives of Computational Methods in Engineering*, 28(7):4549–4571, 2021.

Maurizio Muzzupappa, Loris Barbieri, Fabio Bruno, and Umberto Cugini. Methodology and tools to support knowledge management in topology optimization. *Journal of computing and information science in engineering*, 10(4), 2010.

Nanosoft. nanoCAD, 2022. URL `https://www.nanocad.ru/`.

Aamer Nazir, Ahmed Gohar, Shang-Chih Lin, and Jeng-Ywan Jeng. Flexural properties of periodic lattice structured lightweight cantilever beams fabricated using additive manufacturing: Experimental and finite element methods. *3D Printing and Additive Manufacturing*, 2022.

nTopology, Inc. nTopology, 2022. URL `https://ntopology.com`.

OpenMP. OpenMP, 2022. URL `https://www.openmp.org`.

William Oropallo and Les A Piegl. Ten challenges in 3d printing. *Engineering with Computers*, 32(1):135–148, 2016.

Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

Catherine Pakhomova, Dmitry Popov, Eugenii Maltsev, Iskander Akhatov, and Alexander Pasko. Software for bioprinting. *International Journal of Bioprinting*, 6(3), 2020.

Geetha Palani and Karthik Kannan. Introduction to additive manufacturing for composites: State of the art and recent trends. *High-Performance Composite Structures*, pages 1–24, 2022.

A.A. Pasko, V.V. Pilyugin, and V.N. Pokrovskiy. Geometric modeling in the analysis of trivariate functions. *Computers & Graphics*, 12(3):457–465, 1988. ISSN 0097-8493. doi:https://doi.org/10.1016/0097-8493(88)90070-2. URL `http://www.sciencedirect.com/science/article/pii/0097849388900702`.

Alexander Pasko, Valery Adzhiev, Alexei Sourin, and Vladimir Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The visual computer*, 11(8):429–446, 1995.

Alexander A Pasko, Valery Adzhiev, and Eric Fausett. Multidimensional shape modeling for animation. In *Eurographics (Short Presentations)*, 1999.

Galina I Pasko, Alexander A Pasko, and Tosiyasu L Kunii. Bounded blending for function-based shape modeling. *IEEE Computer Graphics and Applications*, 25 (2):36–45, 2005.

Les A Piegl. Ten challenges in computer-aided design. *Computer-aided design*, 37 (4):461–470, 2005.

Dmitry Popov. 3D level set topology optimization MATLAB code. https://github.com/Dimdimovich/3D_level_set_TopOpt, 2022a. URL https://github.com/Dimdimovich/3D_level_set_TopOpt.

Dmitry Popov. 2d frep topology optimization matlab code, 2022b. URL https://github.com/Dimdimovich/2D_FRep_Topology_Optimization. last accessed 5 August 2022.

Dmitry Popov, Evgenii Maltsev, Alexander Pasko, and Iskander Akhatov. Optimization of a shape and topology optimization for frep models. In *All-Russian Congress on the fundamental problems of theoretical and applied mechanics*, 2019.

Dmitry Popov, Yulia Kuzminova, Evgenii Maltsev, Stanislav Evlashin, Alexander Safonov, Iskander Akhatov, and Alexander Pasko. Structural optimization of frep models and their additive manufacturing. In *CAASE20: The Conference on Advancing Analysis & Simulation in Engineering*, 2020a.

Dmitry Popov, Evgenii Maltsev, Oleg Fryazinov, Alexander Pasko, and Iskander Akhatov. Efficient contouring of functionally represented objects for additive manufacturing. *Computer-Aided Design*, 129:102917, 2020b.

Dmitry Popov, Yulia Kuzminova, Evgenii Maltsev, Stanislav Evlashin, Alexander Safonov, Iskander Akhatov, and Alexander Pasko. CAD/CAM system for additive manufacturing with a robust and efficient topology optimization algorithm based on the function representation. *Applied Sciences*, 11(16):7409, 2021a.

Dmitry Popov, Galkin Timofey, Evgenii Maltsev, and Alexander Pasko. Web CAD/-CAM system for additive manufacturing based on function representation. In *Beam technologies & laser applications*, 2021b.

Mayur Jiyalal Prajapati, Ajeet Kumar, Shang-Chih Lin, and Jeng-Ywan Jeng. Multi-material additive manufacturing with lightweight closed-cell foam-filled lattice structures for enhanced mechanical and functional properties. *Additive Manufacturing*, 54:102766, 2022.

ALR Prathyusha and G Raghu Babu. A review on additive manufacturing and topology optimization process for weight reduction studies in various industrial applications. *Materials Today: Proceedings*, 2022.

progeSOFT. progeCAD, 2022. URL https://www.progesoft.com/products/progecad-professional.

PTC. CREO PARAMETRIC, 2022a. URL https://www.ptc.com/en/products/creo/parametric.

PTC. Onshape, 2022b. URL `https://www.onshape.com/en/`.

Haripriya Rangan, Matthias Ruhl, and Dietmar Saupe. Interactive visualization of implicit surfaces with singularities. *Comput. Graph. Forum*, 16:295–306, 1997.

Singiresu S Rao. *The finite element method in engineering*. Butterworth-heinemann, 2017.

Junuthula Narasimha Reddy. *Introduction to the finite element method*. McGraw-Hill Education, 2019.

Aristides AG Requicha and Herbert B Voelcker. Constructive solid geometry. 1977.

Cartwright Richard, Valery Adzhiev, Alexander Pasko, Yuichiro Goto, and Tosiyasu Kunii. Web-based shape modeling with hyperfun. *IEEE computer graphics and applications*, 25:60–9, 03 2005. doi:10.1109/MCG.2005.49.

Robert McNeel & Associates. Rhino, 2022. URL `https://www.rhino3d.com/`.

S.M. Rump and M. Kashiwagi. Implementation and improvements of affine arithmetic. *Nonlinear Theory and Its Applications, IEICE*, 6(3):341–359, 2015. doi:10.1587/nolta.6.341.

VL Rvachev. *Theory of R-functions and Some Applications*. Naukova dumka Kiev, 1982.

Vladimir L Rvachev, Tatyana I Sheiko, Vadim Shapiro, and I Tsukanov. Transfinite interpolation over implicitly defined sets. *Computer aided geometric design*, 18 (3):195–220, 2001.

Alexander Safonov, Evgenii Maltsev, Svyatoslav Chugunov, Andrey Tikhonov, Stepan Konev, Stanislav Evlashin, Dmitry Popov, Alexander Pasko, and Iskander Akhatov. Design and fabrication of complex-shaped ceramic bone implants via 3d printing based on laser stereolithography. *Applied Sciences*, 10(20):7138, 2020.

Vladimir V Savchenko, Alexander A Pasko, Oleg G Okunev, and Tosiyasu L Kunii. Function representation of solids reconstructed from scattered surface points and contours. In *Computer Graphics Forum*, volume 14, pages 181–188. Wiley Online Library, 1995.

Benjamin Schmitt, Alexander Pasko, and Vladimir Savchenko. Extended space mapping with bezier patches and volumes. In *Implicit Surfaces*, volume 99, pages 25–31, 1999.

Uwe Schramm and Ming Zhou. Recent developments in the commercial implementation of topology optimization. In Martin Philip Bendsøe, Niels Olhoff, and Ole Sigmund, editors, *IUTAM Symposium on Topological Design Optimization of Structures, Machines and Materials*, pages 239–248, Dordrecht, 2006. Springer Netherlands.

T.K. Sellis. Efficiently supporting procedures in relational database systems. In *SIGMOD '87*, pages 278–291, 1987.

Nurettin Sezer, Zafer Evis, and Muammer Koc. Additive manufacturing of biodegradable magnesium implants and scaffolds: Review of the recent advances and research trends. *Journal of Magnesium and Alloys*, 9(2):392–415, 2021.

Linda G. Shapiro. Connected component labeling and adjacency graph construction. 1996.

Vadim Shapiro and I Tsukanov. The architecture of sage–a meshfree system based on rfm. *Engineering with Computers*, 18(4):295–311, 2002.

R. Shih. *Parametric Modeling with Siemens NX (Spring 2022 Edition)*. SDC Publications, 2022. ISBN 9781630575304. URL https://books.google.ru/books?id=BsVqEAAAQBAJ.

Dave Shreiner, Bill The Khronos OpenGL ARB Working Group, et al. *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1*. Pearson Education, 2009.

Lei Shu, Michael Yu Wang, Zongde Fang, Zhengdong Ma, and Peng Wei. Level set based structural topology optimization for minimizing frequency response. *Journal of Sound and Vibration*, 330(24):5820–5834, 2011.

S Shuib, MIZ Ridzwan, AY Hassan, and MN Mohamad Ibrahim. Topology optimisation of hip prosthesis to reduce stress shielding. *WIT Transactions on The Built Environment*, 80, 2005.

Siemens. Solid Edge, 2022a. URL https://solidedge.siemens.com/en/.

Siemens. Siemens NX, 2022b. URL https://www.plm.automation.siemens.com/global/en/products/nx/.

Ole Sigmund. A 99 line topology optimization code written in matlab. *Structural and multidisciplinary optimization*, 21(2):120–127, 2001.

J.M. Snyder. Interval analysis for computer graphics. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 121–130, New York, NY, USA, 1992. ACM. ISBN 0-89791-479-1. doi:10.1145/133994.134024.

SolveSpace contributors. SolveSpace, 2022. URL https://solvespace.com.

Yanzhi Song, Zhouwang Yang, Yuan Liu, and Jiansong Deng. Function representation based slicer for 3d printing. *Computer Aided Geometric Design*, 62:276–293, 2018.

J. Stolfi and L. H. de Figueiredo. *Self-Validated Numerical Methods and Applications*. IMPA, 1997.

J. Stolfi and L.H. de Figueiredo. An Introduction to Affine Arithmetic. *TEMA Tend. Mat. Apl. Comput.*, 4(3):297–312, 2003. ISSN 2179-8451. doi:https://doi.org/10.5540/tema.2003.04.03.0297. URL `https://tema.sbmac.org.br/tema/article/view/352`.

N. Stolte and A. Kaufman. Parallel spatial enumeration of implicit surfaces using interval arithmetic for octree generation and its direct visualization. In *Implicit Surfaces'98 Proceedings*, pages 81–88, January 1998. URL `https://pdfs.semanticscholar.org/7a21/36d8222f6c19c5bbc4e30598038bdb75f623.pdf?_ga=2.238635461.496188997.1563888038-540992412.1562229864`.

Ian Stroud. *Boundary representation modelling techniques*. Springer Science & Business Media, 2006.

Bjarne Stroustrup. What is object-oriented programming? *IEEE software*, 5(3): 10–20, 1988.

Teruo Sunaga. Theory of an interval algebra and its application to numerical analysis. *Japan Journal of Industrial and Applied Mathematics*, 26:125–143, 2009.

Ivan E Sutherland. Sketchpad a man-machine graphical communication system. *Simulation*, 2(5):R–3, 1964.

Akihiro Takezawa, Shinji Nishiwaki, and Mitsuru Kitamura. Shape and topology optimization based on the phase field method and sensitivity analysis. *Journal of Computational Physics*, 229(7):2697–2718, 2010.

Nathanael Tan and Richard J van Arkel. Topology optimisation for compliant hip implant design and reduced strain shielding. *Materials*, 14(23):7184, 2021.

Taylor Otwell. Lumen, 2022. URL `https://lumen.laravel.com/`.

Alexander Tereshin, Eike Falk Anderson, Alexander A Pasko, and Valery Adzhiev. Space-time blending for heterogeneous objects. In *Eurographics (Short Papers)*, pages 45–48, 2020.

Alexander Tereshin, Alexander Pasko, Oleg Fryazinov, and Valery Adzhiev. Hybrid function representation for heterogeneous objects. *Graphical Models*, 114:101098, 2021.

The FreeCAD Team. FreeCAD, 2022. URL `https://www.freecadweb.org`.

The Khronos Group Inc. WebGL, 2022. URL `https://www.khronos.org/webgl/`.

The LibreCAD Team. LibreCAD, 2022. URL `https://librecad.org/`.

The MathWorks, Inc. MATLAB, 2022. URL `https://www.mathworks.com/`.

Thomas Theußl, Torsten Möller, and Eduard Gröller. Optimal regular volume sampling. *Proceedings Visualization, 2001. VIS '01.*, pages 91–546, 2001.

Xiaojie Tian, Qingyang Wang, Guijie Liu, Yunxiang Liu, Yingchun Xie, and Wei Deng. Topology optimization design for offshore platform jacket structure. *Applied Ocean Research*, 84:38–50, 2019.

Tim Berners-Lee. HTML5 Documentation, 2022. URL https://html.spec.whatwg.org/multipage/.

Top Systems. T-FLEX CAD, 2022. URL https://www.tflexcad.ru.

H. Uchibori. Slicing function-based shape models for rapid prototyping. Master's thesis, Tokyo: Hosei University, 2004.

Ultimaker. Curaengine, 2013. URL https://github.com/Ultimaker/CuraEngine. Last accessed 19 February 2019.

VariCAD. VariCAD, 2022. URL https://www.varicad.com/en/home/products/products/.

Vectorworks, Inc. Vectorworks, 2022. URL https://www.vectorworks.net/.

Vizerra SA. Revizto, 2022. URL https://revizto.com/.

Daria Vlah, Roman Žavbi, and Nikola Vukašinović. Evaluation of topology optimization and generative design tools as support for conceptual design. In *Proceedings of the design society: DESIGN conference*, volume 1, pages 451–460. Cambridge University Press, 2020.

Panagiotis Vogiatzis, Shikui Chen, Xiao Wang, Tiantian Li, and Lifeng Wang. Topology optimization of multi-material negative poisson's ratio metamaterials using a reconciled level set method. *Computer-Aided Design*, 83:15–32, 2017.

X.-H. Vu, D. Sam-Haroud, and B. Faltings. Combining multiple inclusion representations in numerical constraint propagation. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 458–467, November 2004. doi:10.1109/ICTAI.2004.40.

Jiayi Wang, Santosh Reddy Sama, Paul C Lynch, and Guha Manogharan. Design and topology optimization of 3d-printed wax patterns for rapid investment casting. *Procedia Manufacturing*, 34:683–694, 2019.

Lvdi Wang, Yizhou Yu, Kun Zhou, and Baining Guo. Multiscale vector volumes. *ACM Trans. Graph.*, 30(6):1–8, December 2011. doi:10.1145/2070781.2024201.

Michael Yu Wang, Xiaoming Wang, and Dongming Guo. A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering*, 192(1-2):227–246, 2003.

SY Wang, Kian Meng Lim, Boo Cheong Khoo, and Michael Yu Wang. An extended level set method for shape and topology optimization. *Journal of Computational Physics*, 221(1):395–421, 2007.

W. Warmus. Calculus of approximations. *Bull. Acad. Polon. Sci. Cl. III.*, 4:253–259, 1956.

Peng Wei, Michael Yu Wang, and Xianghua Xing. A study on x-fem in continuum structural optimization using a level set model. *Computer-Aided Design*, 42(8): 708–719, 2010.

Peng Wei, Zuyu Li, Xueping Li, and Michael Yu Wang. An 88-line matlab code for the parameterized level set method based topology optimization using radial basis functions. *Structural and Multidisciplinary Optimization*, 58(2):831–849, 2018.

Peng Wei, Yang Yang, Shikui Chen, and Michael Yu Wang. A study on basis functions of the parameterized level set method for topology optimization of continuums. *Journal of Mechanical Design*, 143(4):041701, 2021.

P. Wen, W. Xiaojun, T. Gao, and C. Wu. Kd-tree based ols in implicit surface reconstruction with radial basis function. In *International Conference on Artificial Reality and Telexistence: Advances in Artificial Reality and Tele-Existence*, volume 4282, pages 861–870, 2006. doi:10.1007/11941354_89.

G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The Visual Computer*, pages 227–234, August 1986.

Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, 2:227–234, 2005.

Qi Xia, Tielin Shi, and Liang Xia. Topology optimization for heat conduction by combining level set method and beso method. *International Journal of Heat and Mass Transfer*, 127:200–209, 2018.

Zhaohui Xia, Yingjun Wang, Qifu Wang, and Chao Mei. Gpu parallel strategy for parameterized lsm-based topology optimization using isogeometric analysis. *Structural and Multidisciplinary Optimization*, 56(2):413–434, 2017.

Kentaro Yaji, Takayuki Yamada, Seiji Kubo, Kazuhiro Izui, and Shinji Nishiwaki. A topology optimization method for a coupled thermal–fluid problem using level set boundary expressions. *International Journal of Heat and Mass Transfer*, 81: 878–888, 2015.

Xiangyu You. *Differential equation-based shape interpolation for surface blending and facial blendshapes.* PhD thesis, Bournemouth University, 2022.

Minghao Yu, Shilun Ruan, Xinyu Wang, Zheng Li, and Changyu Shen. Topology optimization of thermal–fluid problem using the mmc-based approach. *Structural and Multidisciplinary Optimization*, 60(1):151–165, 2019.

Danwei Zhang, Wei Yang Samuel Lim, Solco Samantha Faye Duran, Xian Jun Loh, and Ady Suwardi. Additive manufacturing of thermoelectrics: Emerging trends and outlook. *ACS Energy Letters*, 7:720–735, 2022a.

Longfei Zhang, Shengfa Wang, Baojun Li, Yi Wang, Zhongxuan Luo, and Ligang Liu. Function representation based analytic shape hollowing optimization. *Computer-Aided Design*, 144:103156, 2022b.

Shiwei Zhou and Qing Li. A variational level set method for the topology optimization of steady-state navier–stokes flow. *Journal of Computational Physics*, 227 (24):10178–10195, 2008.

Olek C Zienkiewicz, Robert Leroy Taylor, and Jian Z Zhu. *The finite element method: its basis and fundamentals*. Elsevier, 2005.

ZWSOFT CO., LTD. ZW3D, 2022. URL https://www.zwsoft.com/product/zw3d.

Appendix A

Interval and Affine Arithmetic

A.1 Interval Arithmetic

Denote the interval as $I = [a, b] \in \mathbb{R}$; it is a set of real numbers that are located between two numbers a and b, which are also included in the set: $[a, b] := \{x \in \mathbb{R} \mid a \leq x \leq b\ \}$,

a – lower bound of the interval, $\inf_I = a$;

b – upper bound of the interval, $\sup_I = b$.

Interval arithmetic operations:

$$a + b = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$$

$$a - b = [\underline{a} - \overline{b}, \overline{a} - \underline{b}]$$

$$a * b = [\min\{\underline{ab}, \overline{a}\underline{b}, \overline{b}\underline{a}, \overline{a}\overline{b}\}, \max\{\underline{ab}, \overline{a}\underline{b}, \overline{b}\underline{a}\overline{b}\}]$$

$$a/b = a * \left[\frac{1}{\overline{b}}, \frac{1}{\underline{b}}\right]\ for\ b \notin 0.$$

Middle point or centre of the interval:

$$mid(I) = \frac{a+b}{2}.$$

Radius of the interval:

$$rad(I) = \frac{a-b}{2}.$$

Denote the natural interval extension of the function $f(x, y, z)$ as $F(X, Y, Z)$, where $X, Y, Z \in \mathbb{R}$; in accordance with the fundamental theorem of interval analysis, it can be concluded that:

$$\{f(x, y, z) \mid x \in X, y \in Y, z \in Z\} \subseteq F(X, Y, Z).$$

Thus, the result of interval estimation $F(X, Y, Z)$ contains a set of values of the function $f(x, y, z)$ in the cell (X, Y, Z). This fundamental inclusion property allows the user to calculate the upper and lower bounds of the estimation of the intervals of function values [Moore, 1966], [Snyder, 1992].

Transcendental functions with arithmetic operations are used in 3D modelling. One can construct the interval extension for most such functions: e^x, $\ln x$, $\sqrt{x}$, $\sin x$, and $\cos x$. An example of interval extension for the monotonous decay function is:

$$e^{-X} = [e^{-\overline{x}}, e^{-\underline{x}}].$$

For other functions, the interval extension can be constructed by dividing function into monotonic (decay or increasing) intervals [Stolte and Kaufman, 1998]:

$$X^n = \begin{cases} [\underline{x}^n, \overline{x}^n] & n\, odd\, or\, \underline{x} \geq 0 \\ [\overline{x}^n, \underline{x}^n] & n\, even\, or\, \underline{x} \leq 0 \\ [0, \max(-\underline{x}, \overline{x})] & n\, even\, \underline{x} < 0 < \overline{x} \end{cases}$$

To ensure the inclusion property, it is necessary to take into account the rounding modes of the arithmetic operations, which depend on the types of the variables. For instance, the Boost library requires selecting the rounding policy according to the data types used (integer or float) [Brönnimann et al., 2006]. It is worth noting that one way to increase the performance of the interval estimation calculation is to reduce the number of switches of the rounding mode of the floating-point unit (FPU). For example, the calculation of the sum of the intervals $[a, b] + [c, d] = [\underline{(a + c)}, \overline{(b + d)}]$ requires changing the rounding mode (towards $+\infty$ and $-\infty$); however, one can use only one mode using the replacement operation:

$\underline{(a+c)} = -\overline{(-a-c)}$. This leads to an increase in the speed of the calculation, because the operation of changing the sign is cheaper than the operation of changing the rounding mode.

A.2 Affine Arithmetic

In affine arithmetic, a partially unknown quantity x is represented by affine forms $\hat{x}$ (i.e., first-degree polynomials):

$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + ... + x_n\varepsilon_n,$$

where the x_i are finite real numbers and the ε_i are symbolic unknown real-valued variables located within the interval U = [-1;1]. The coefficient x_0 is called the central value of the affine form $\hat{x}$, the coefficients x_i are called the partial deviations, and the ε_i are called the noise symbols. The components ε_i are independent. On the implementation level, this means that each of them corresponds to an initialized variable if it was not initialized as an affine combination of other variables. Let us represent the quantity x with the affine form $\hat{x}$ as above; then, the interval bounds for x will follow $x \in [x_0 - r_x, x_0 + r_x]$, where $r_x = \sum_{i=1}^{n} |x_i|$ is called the total deviation of $\hat{x}$. For estimation of the quantity x, the same methods are used as for interval arithmetic. The affine arithmetic extension is constructed for the quantity x via replacement of all elementary operations and functions by their affine forms. The computations are conducted under this affine extension. For the representation of the affine operation, let us consider two quantities x and y represented by the affine:

$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + ... + x_n\varepsilon_n$$

$$\hat{y} = y_0 + y_1\varepsilon_1 + y_2\varepsilon_2 + ... + y_n\varepsilon_n.$$

Then, the operation z between these quantities x and y can be written as follows:

$$z = f(x, y) = f(x_0 + \sum_{i=1}^{n} x_i\varepsilon_i, y_0 + \sum_{i=1}^{n} y_i\varepsilon_i) = f^*(\varepsilon_1, ..., \varepsilon_n), \varepsilon_i \in U,$$

where $f^*(\varepsilon_1, ..., \varepsilon_n)$ is a function $U^n \to \mathbb{R}$.

If the values of x are y are partially dependent, then the affine forms of these quantities have shared noise symbols. This means that the joint interval Z of x and y is not a rectangle $R = XY$ as in interval arithmetic but is a polygon in $\mathbb{R}^2$. Therefore, the bound intervals of quantities produced in affine arithmetic can be better than those in interval arithmetic.

If f^* is linear, then $\hat{y}$ can be represented easily:

$$\hat{x} + \hat{y} = (x_0 + y_0) + (x_1 + y_1)\varepsilon_1 + ... + (x_n + y_n)\varepsilon_n$$

$$\hat{x} - \hat{y} = (x_0 - y_0) + (x_1 - y_1)\varepsilon_1 + ... + (x_n - y_n)\varepsilon_n$$

$$\alpha\hat{x} = (\alpha x_0) + (\alpha x_1)\varepsilon_1 + ... + (x_n - (\alpha x_n)\varepsilon_n, \alpha \in \mathbb{R}.$$

If f^* is not linear, then an approximated linear function f^a for f^* with introduced approximated error is used. Every non-affine operation leads to the addition of an extra term $z_{N+1}\varepsilon_{N+1}$:

$$\hat{z} = f^a(\varepsilon_1, ..., \varepsilon_n) + z_{N+1}\varepsilon_{N+1} = z_0 + z_1\varepsilon_1 + z_2\varepsilon_2 + ... + z_n\varepsilon_n + z_{N+1}\varepsilon_{N+1},$$

where f^a approximates a non-affine operation, z_{N+1} is an upper bound on the absolute magnitude of the approximation error and N is the number of noises before the considered operation. A new term $z_{N+1}\varepsilon_{N+1}$ is used to represent the difference between f^* and f^a.

The computation of the affine arithmetic form requires a good affine approximation f^a for each simple non-affine operation f^*. Different approximation techniques are discussed in [Stolfi and de Figueiredo, 1997] for the affine forms of several functions: Chebyshev, Min-range and Interval approximation.

For example, here is a classic example for multiplication in the AA form [de Figueiredo and Stolfi, 2004]:

$$\hat{x}\hat{y} = x_0 y_0 + \sum_{i=1}^{n} (x_0 y_i + y_0 x_i)\varepsilon_i + z_{N+1}\varepsilon_{N+1},$$

where

$$z_{N+1} \geq \Big| \sum_{i=1}^{n} x_i \varepsilon_i \sum_{i=1}^{n} y_i \varepsilon_i \Big|, \varepsilon_i \in [-1; 1].$$

In addition, Stolfi and de Figueeirdo suggest introducing an extra noise symbol for roundoff errors during calculations.

For a more detailed comparison of IA and AA see [de Figueiredo and Stolfi, 2004], section 4. Affine arithmetic and the dependency problem.

Appendix B

The implicit surfaces tested

Sphere: $f = R^2 - x^2 - y^2 - z^2$, where $R = 6$.

Decocube: $f = \big((x^2+y^2-0.82)^2+(z^2-1)^2\big)\big((x^2+z^2-0.82)^2+(y^2-1)^2\big)\big((z^2+y^2-0.82)^2+(x^2-1)^2\big)-1$.

Orthocircle: $f = \big((x^2+y^2-1)^2+z^2\big)\big((y^2+z^2-1)^2+x^2\big)\big((z^2+x^2-1)^2+y^2\big) - a^2\big(1+(x^2+y^2+z^2)\big)$; $a = 0.075, b = 3$.

Union of spheres: $f = s_1 \bigcup s_2$, where $s_1 = 2^2 - x^2 - y^2 - z^2, s_1 = 2^2 - (x-2)^2 - (y-2)^2 - z^2$.

The construction tree of the cylinder with lattice model:

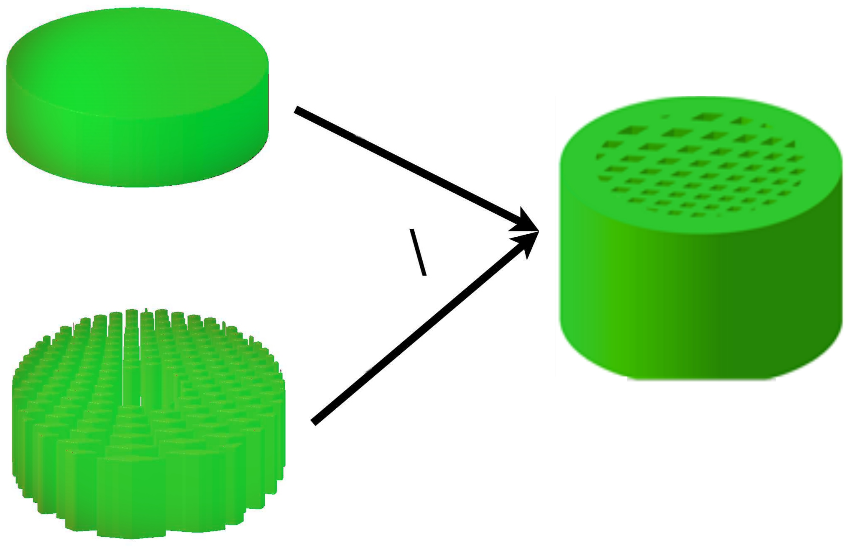


Figure B-1: FRep tree for the cylinder model

The construction tree of the gear model:

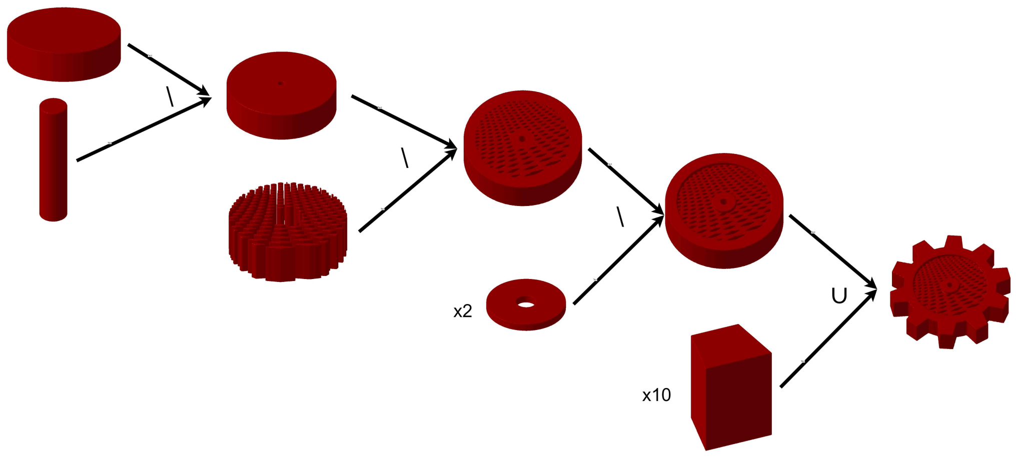


Figure B-2: FRep tree for the gear model

The microstructure model in the HyperFun [Richard et al., 2005] syntax:

Function my_model(x, a)

```
$R \leftarrow 16;$
$thickness \leftarrow 20;$
$l \leftarrow 0.5;$
$distance \leftarrow sqrt(x[1] * x[1] + x[2] * x[2]);$
$freqX \leftarrow 10; freqY \leftarrow 10; freqZ \leftarrow 10;$
$sx \leftarrow sin(freqX * x[1]) - l;$
$sy \leftarrow sin(freqY * x[2]) - l;$
$sz \leftarrow sin(freqZ * x[3]) - l;$
$rx \leftarrow sy \ \& \ sz;$
$ry \leftarrow sx \ \& \ sz;$
$rz \leftarrow sx \ \& \ sy;$
$lattice \leftarrow (rx|ry|rz);$
$big\_cylinder \leftarrow R * R - x[1] * x[1] - x[2] * x[2];$
$small\_cylinder \leftarrow big\_cylinder - thickness;$
$inter \leftarrow small\_cylinder \ \& \ lattice;$
$sub \leftarrow big\_cylinder \setminus small\_cylinder;$
$sub \leftarrow sub \mid inter;$
$my\_model \leftarrow sub;$
```