Efficient contouring of functionally represented objects for additive manufacturing

Dmitry Popov[1], Evgenii Maltsev[1], Oleg Fryazinov[2], Alexander Pasko[1], Iskander Akhatov[1]
[1]Skolkovo Institute of Science and Technology, Russia
[2]Bournemouth University, UK,

Abstract

Functionally (implicitly) defined 3D objects allow us to quite easily model parts with complex topology such as lattices and organic-like structures with a high level of flexibility. Previous works in this area are based on the direct generation of CNC programs for the 3D printing of these objects and are backed by the growing support for this input format from hardware manufacturers. Efficient contouring of functionally defined models, however, is not an easy task. In this paper, we develop an algorithm for contour extraction of implicitly defined objects for direct additive manufacturing (AM). By comparing various adaptive and exhaustive (non-adaptive) methods of the function representation contouring for AM (FRepCAM), we make a set of recommendations for its usage depending on the specific resolution of the printer. In particular, we use a novel criterion based on affine arithmetic to maintain efficiency while preserving the robustness of the contouring process. The techniques mentioned were evaluated for algebraic and non-algebraic solids and heterogeneous models under a resolution that is comparable with that of current AM technology. The results show that the chosen adaptation criteria allow us to efficiently obtain a contour for complex models and generally outperform those of traditional algorithms based on exhaustive enumeration, especially for high-resolution contouring. In addition, the results present proof of the printability of implicitly defined objects with different 3D printing techniques.

Keywords: Function Representation; FRep; Implicit Representation; Level

Set; Interval Arithmetic; Affine Arithmetic; Rapid Prototyping; Additive Manufacturing; Direct Fabrication; Adaptive Contouring; Process Planning.

1. Introduction

Over the past decade, the area of additive manufacturing (AM) has shown exponential growth with new processes and materials for 3D printing. Moreover, the hardware for AM has rapidly developed, with dozens of new models of 3D printers going to the market every year and with the machines becoming increasingly sophisticated on the one hand and yet user-friendly on the other. At the same time, it can be seen that the software for design, engineering and manufacturing cannot catch up with the rapid development of the hardware side of AM. One of the main reasons for slower growth is that AM uses geometric representations traditional for computer graphics rather than those specific to the AM area.

Modern geometric modelling software supports more than sixty file formats of 3D models, with the majority being represented with boundary surfaces in a representation called boundary representation (BRep). Likewise, the main format for geometry in AM is stereolithography (STL) [1], where the solid object is represented by a set of boundary polygons. New AM-aimed formats .AMF and .3MF improve STL with several useful additions, such as colour and material information. Nevertheless, they inherit major drawbacks from STL, such as the difficulty of obtaining arbitrary precision for the resulting 3D printed object and very limited support of multi-material models and, in general, heterogeneous objects. Recently, voxel representation was introduced in AM to overcome the problem of heterogeneous modelling, but at the same time, this representation is not able to overcome the problem of large size for complex models with high precision. This leads to the use of procedural approaches for dynamic generation of voxel models with a given accuracy [2].

All rapid prototyping techniques use a so-called 2½ representation of models. One of the standards that describes a model as a set of ordered contours is the

common layer interface (CLI) [3]. Another 2½ format is the SLiCe format (SLC) [4]. Both of these standards require storing sets of polylines for several z-levels.

In this paper, we focus on geometric representation in an implicit form, where the object is represented by a mathematical function. This representation, called function representation (FRep), has recently shown significant potential for AM compared to traditional methods. For example, it allows the modelling of parts with complex topologies, such as lattices and organic-like structures, quite easily and with a high level of flexibility. Moreover, this representation allows us to define the heterogeneous nature of objects conveniently with the definition of the geometry, and the resulting geometric model becomes a closed volume with guaranteed printability. Instead of storing a mesh as a set of polygonal approximation formats (STL), voxels or polylines, FRep stores only its implicit functional description as a symbolic expression and requires less memory. This representation covers standard constructive solid geometry (CSG) [5], which considers simple solids (primitives) as elementary units of the model that can be connected by applying set-theoretic operations to primitives. FRep supports the concept of direct manufacturing where no intermediate formats (such as .STL, .AMF, .3MF) are needed for the fabrication of designed models. On the other hand, one needs to perform calculation to "extract" the model from the implicit form. This feature allows us to fit the model into a discrete representation with arbitrary precision. A more detailed discussion can be found in the implicit modeling chapter of [6].

We describe algorithms and methods for contour extraction of 3D objects represented by FRep without explicit surface generation and thereby convert converting FRep into a 2½ model ready for direct additive manufacturing. Function representation contouring for AM (FRepCAM) is a necessary part of the AM chain for FRep models. Triangulated models with a complex topology can occupy up to several GBs of disk space in STL format, and this size increases with increasing accuracy or complexity. There are a few 3D printers that work with such large files, and it takes days to pre-process them. One of the possible solutions is the concept of direct AM demonstrated in [7]. Direct

manufacturing [8] assumes no intermediate geometric representations between the modelling system and the CNC program. Such a concept can be included in the development of a computer-aided design (CAD) system with a geometric modelling kernel based on FRep, which includes computer-aided manufacturing (CAM) modules. According to [9], such a system can carry out the steps of product design, model creation and generation of the build file for the AM digital thread (Figure 1). Moreover, the tessellated model becomes an unnecessary artefact of that chain.

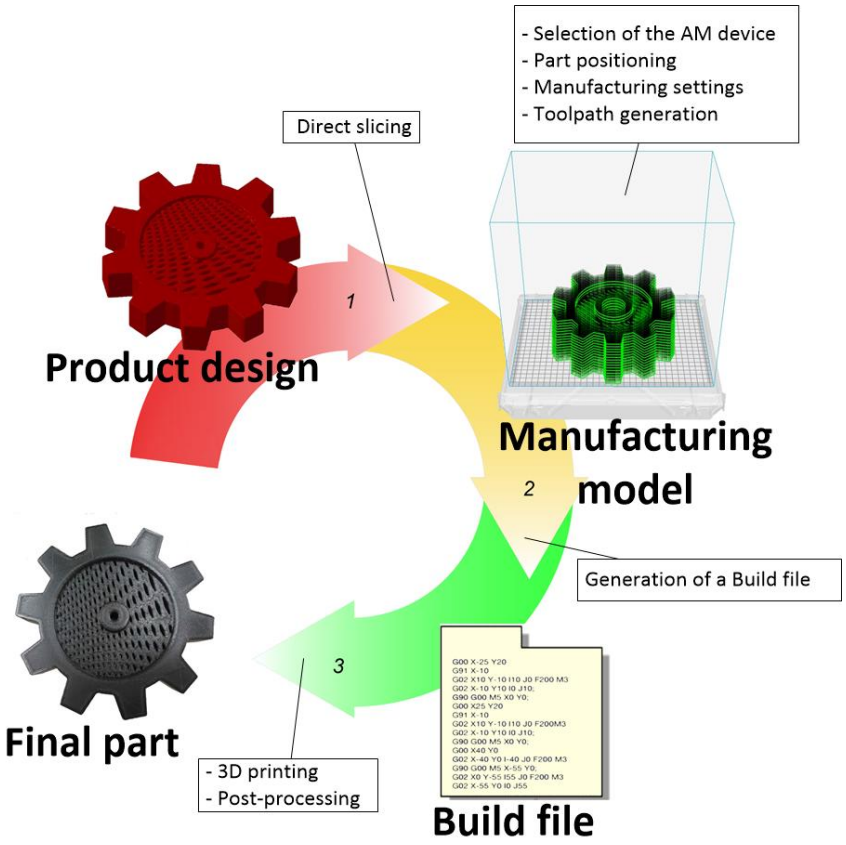


Figure 1: AM digital thread without model tessellation

Generally, AM uses computing machines similar to a computer numerical

control (CNC) machine; therefore, pre-processing methods for printing of 3D models depend on the chosen strategy and the capabilities of this equipment. In particular, the printing strategy depends on the number of available axes of the device. This paper considers the techniques of contouring and obtaining 2D slices of functionally represented 3D models for 3-axis AM devices. In particular, we focus on different techniques of contouring, such as contouring with a regular grid or exhaustive enumeration and adaptive contouring with interval and affine arithmetic.

Above mentioned techniques were evaluated for algebraic and non-algebraic solids at a resolution that is appropriate for the current state of AM technologies. The results show that the chosen adaptation criteria allow us to outperform traditional contouring algorithms based on exhaustive enumeration for complex models. The most dramatic difference appears in contouring with high resolution. In addition, the results present proof of the printability of the designed implicitly defined objects with different 3D printing techniques.

The contributions of this work are as follows:

1. We revisit the concept of direct manufacturing [8] with geometry represented by FRep by using contour slices as in [7] and preparing GCODE subroutines for two different 3D printers, where the slices are generated directly from FRep objects without an intermediate step of surface generation.
2. We compare various adaptive and exhaustive (non-adaptive) methods of FRepCAM, including exhaustive enumeration and reliable methods based on interval arithmetic.
3. We present a novel criterion based on affine arithmetic that reduces the number of cells processed during the adaptive subdivision algorithm for contour extraction while preserving the robustness of the algorithm.
4. Based on the comparison of the methods for contour extraction of FRep curves, we make a set of recommendations for their usage depending on the specific resolution of the printer.

5. We present the results of contouring with a resolution of up to 0.5 μm, which is lower than the minimal resolution of 400 μm presented in [7].

2. Background and related works

2.1. Function representation

We first describe the general concepts of FRep. Following [10], the key parts of the modelling process with FRep can be formalized as a triple (M, Φ, W), where M is a set of geometric objects, Φ is a set of geometric operations, and W is a set of relations for the set of objects. In this paper, we focus on M. Geometric objects are closed subsets of an n-dimensional space E^n. For our purposes of 3D modelling and 3D printing, we consider E^3. The 3D object is defined in FRep by an inequality:

$$f(x, y, z) \geq 0, \tag{1}$$

where x,y,z are real-valued Cartesian coordinates of a point and f is a real continuous scalar function defined on E^3, also called a defining function. The equation:

$$f(x, y, z) = 0,$$

which defines the boundary of such an object, is called an implicit surface. The function f can be defined by a formula, an evaluation algorithm or tabulated values with an appropriate interpolation procedure. For each point $P = (x, y, z) \in E^3$:

$$\begin{aligned} &f(P) > 0 \text{ if } P \text{ is inside the object;} \\ &f(P) = 0 \text{ if } P \text{ is on the boundary of the object;} \\ &f(P) < 0 \text{ if } P \text{ is a point outside the object.} \end{aligned} \tag{2}$$

The majority of modelling systems that work with FRep employ an approach similar to that of CSG. A designer can introduce or write down several simple

functions for primitives and then construct more complex objects by using operations on these primitives. FRep supports many traditional CAD operations, for example, set-theoretic operations (realized via R-functions) [11], blending, tapering, offsetting [10] and even filleting and rounding [12]. Additionally, one can invent a sophisticated defining function $f(x, y, z) = 0$ from scratch. It can describe a new complex geometry or include a new operation. The technique of tessellating a unit cell in [13] or reconstructing the geometric primitive from a point cloud [14] are examples of such user-defined functions. For AM, the FRep approach has the following additional advantages:

1. Point membership classification is a natural operation in FRep; therefore, slice rasterization for image stack-based 3D printers is very simple and efficient.
2. The 3D model can be evaluated with arbitrary quality and with the required level of accuracy, which satisfies the capabilities of the 3D printing equipment.
3. The representation provides a compact description of 3D models while representing smooth models as well as CAD objects with sharp features by using the same approach. Both can be represented by defining functions with the same complexity.
4. Multi-material objects can be defined by a similar function, and therefore FRep can provide a description not only of the geometry but also of materials in the same model.

2.2. Applications of function representation in additive manufacturing

Recently, FRep has attracted increasing attention from AM specialists. One of the earliest works in this field is [15], where the authors considered it as a way to print artworks of the Dutch artist M.C. Escher. In that work, FRep had to be converted to BRep with the STL file format. Limitations of the STL and BRep formats when working with FRep to model the vascular tree were discussed in [16]. In [17], FRep was mentioned as a suitable approach to represent the

complex geometry of absorbing structures. These structures can have different lattice patterns and different skin configurations. In this case, BRep becomes a limiting factor of the size and complexity of their design. The authors of [18] considered FRep as a possible technique for the design of porous structures such as bones while still using BRep as the main geometric representation. The discussion about complex structures was continued in [19]. The newer work [20] considers FRep as a suitable technique for scaffold modelling. Bodkin, Bibb, and Harris [21] discussed some problems of prosthetic design. They mentioned FRep as a part of their own design strategy and noted an existing bottleneck with intermediate STL representation. The authors of [2] used functions to insert graded lattices and other cellular structures into models described via voxels. FRep was considered a possible technique for functionally graded material modelling in [22] and [23]. [24], [25] and [26] show how to fill a model with a cellular structure based on a chosen pattern, but again the models have to be converted from FRep to STL to handle 3D printing tasks. Recently, FRep was used for 3D printing purposes in [7]. The authors demonstrated the pipeline from FRep modelling to 3D printing with a self-generated G-code file. They used the marching squares (MS) algorithm [27] and interval arithmetic during the contouring process. At the same time, other important aspects, such as the problem with ambiguous cells in the contouring process and the practicality of applying adaptive algorithms, were not considered.

3. Contouring the model

In this work, we use the 2½ representation of models or slices to generate the toolpath of the 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. For geometric models with FRep (see equation (1)), the set of contours of an object

on level $z = z_0$ can be formally defined as the 2D point set $H = \{(x, y) \mid f(x, y, z_0) = 0\}$. In our text, we denote 2D objects built of contours on level z_0 as $f_{z_0}(x, y) = f(x, y, z_0)$ so that we can use equation (2) with $P = (x, y)$ and the 2D function f_{z_0} for the point classification-process. 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 [28], but in this work, we only consider 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 FRepCAM, therefore, is to efficiently approximate the set H given the resolution of the target AM hardware, ensuring that all the contours are included in the toolpath for the current layer.

3.1. 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 [29],[30],[31].
2. Adaptive subdivision of the space on adjacent cells [32],[33],[34].
3. Surface tracking or numerical continuation [35],[36],[37].

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 [38]. 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 [39]. Because of the described limitation, this method will be excluded from discussion. Below, we discuss the conventional methods from the first two categories.

3.1.1. Contour extraction on a regular grid

The methods of contour extraction on a regular grid, mainly based on the marching cubes (marching squares in 2D) algorithm [29], 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. In 2D, there are cells of 16 types (Figure 2) 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.

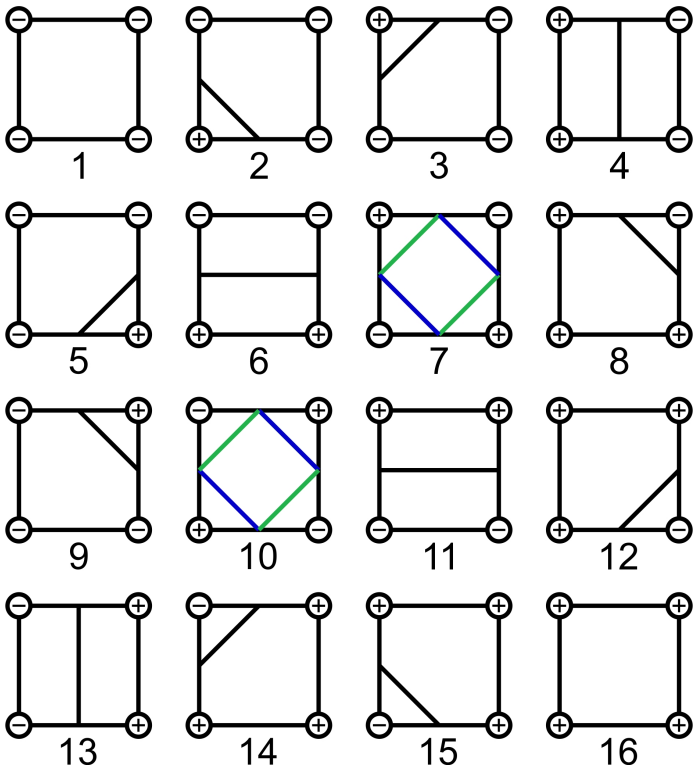


Figure 2: Look-up table for marching squares

The process of contour creation, therefore, is to process the cells and to connect the resulting segments accordingly. 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 [40], 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 is to calculate the additional value of the function at the centre of a cell [41], [42]. An example of resolving ambiguity inside cell number 7 is illustrated in Figure 3. If the function value of the cell centre is positive, then this cell has the type

of Figure 3a; otherwise, the cell has the type of Figure 3b. 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.

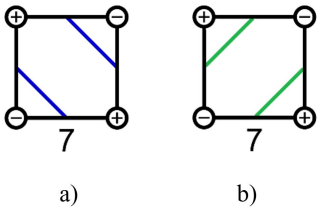


Figure 3: Variants of ambiguity resolution inside cell number 7

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

$$h(x,y) = f_{z_0}(x_1,y_1)\frac{x_2-x}{x_2-x_1} + f_{z_0}(x_2,y_1)\frac{x-x_1}{x_2-x_1}\frac{y_2-y}{y_2-y_1} + \\ + f_{z_0}(x_1,y_2)\frac{x_2-x}{x_2-x_1} + f_{z_0}(x_2,y_2)\frac{x-x_1}{x_2-x_1}\frac{y-y_1}{y_2-y_1} \quad (3)$$

for the considered cell (Figure 4). Here, (x_1,y_1) is the lower left and (x_2,y_2) is the upper right vertices of the cell. This approach is described in [43] 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:

$$x_c = \frac{x_2 f_{z_0}(x_1,y_1) + x_1 f_{z_0}(x_2,y_2) - x_1 f_{z_0}(x_2,y_1) - x_2 f_{z_0}(x_1,y_2)}{f_{z_0}(x_2,y_2) + f_{z_0}(x_1,y_1) - f_{z_0}(x_1,y_2) - f_{z_0}(x_2,y_1)}, \quad (4)$$

$$y_c = \frac{y_2 f_{z_0}(x_1,y_1) + y_1 f_{z_0}(x_2,y_2) - y_1 f_{z_0}(x_1,y_2) - y_2 f_{z_0}(x_2,y_1)}{f_{z_0}(x_2,y_2) + f_{z_0}(x_1,y_1) - f_{z_0}(x_1,y_2) - f_{z_0}(x_2,y_1)}. \quad (5)$$

In addition, it can be seen that for ambiguous cells with $x_1 \le x_c \le x_2$ and $y_1 \le y_c \le y_2$, two possible configurations of the hyperbola are possible. For example, for $y = y_1$, one can obtain $x_0 \le x_c$ or $x_c \le x_0'$, where the value of the

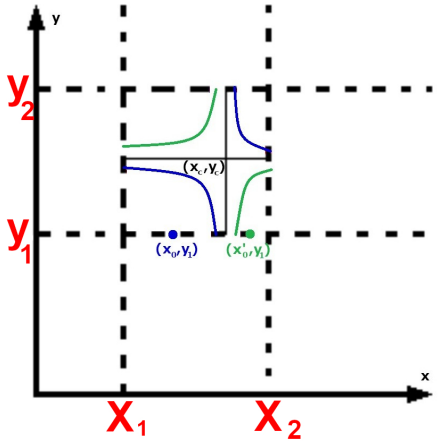


Figure 4: Bilinear approximation of the defining function.

defining function is zero. The first option leads to the "blue" cell configuration and the second to the "green" configuration (Figure 3). 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.

3.1.2. 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 [38].

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 5(a-d). The equations for the functionally defined model for these pictures are presented in Appendix B.

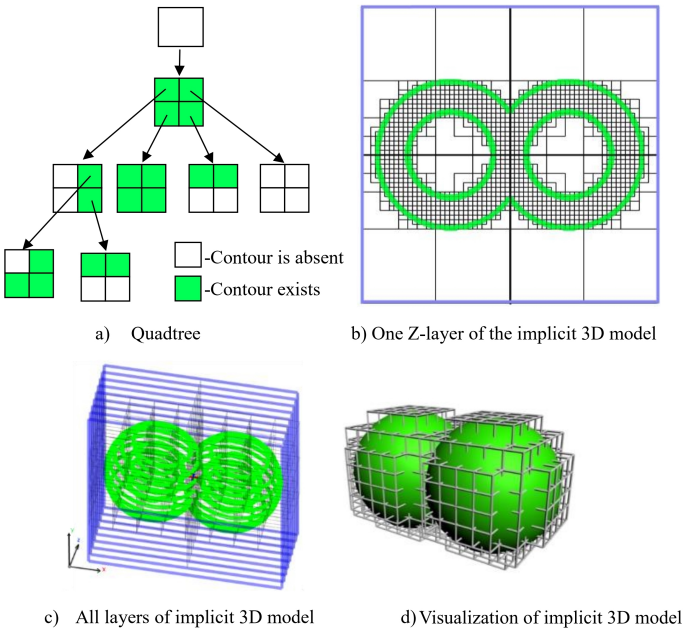


Figure 5: The process of contour extraction using the quadtree

The simplest approach to build an adaptive subdivision is to use the MS algorithm with rough resolution to understand the topology of the implicit curve. The MS 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; examples of these are shown in Figure 6 [38], [44].

Another method to identify the cells that need to be subdivided is to use interval arithmetic (IA). IA is an extension of real arithmetic defined on the set

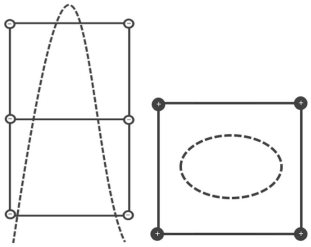


Figure 6: Errors of the marching squares algorithm

of real intervals [45],[46],[47]. 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 [48], [33]. An example of an interval estimation for a function with one variable is shown in Figure 7.

The FRepCAM of the model (a union of spheres) using the IA criteria is shown in Figure 8. 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 [49].

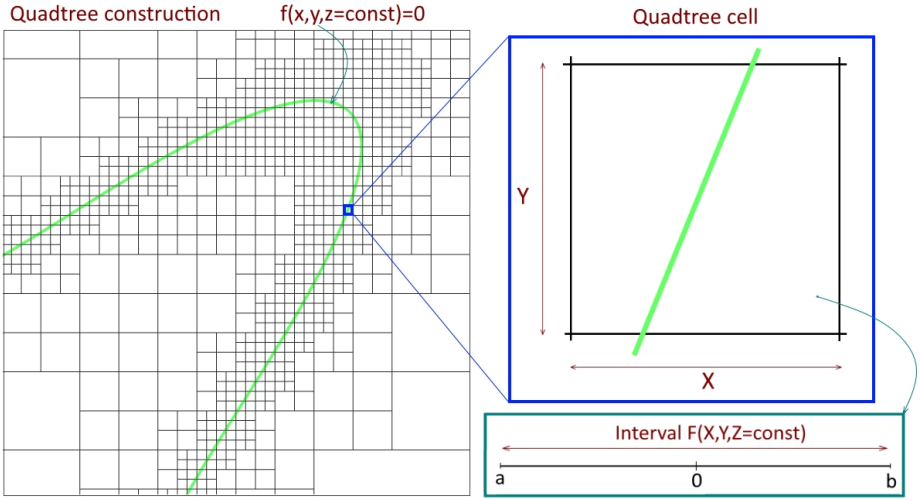


Figure 7: An example of an interval estimation. *f(x,y,z=const)=0* is the contour (green color); The estimation of the function value *f(x,y,z=const)* in the quadtree cell with intervals X and Y is the calculated interval of F(X,Y,Z=const). *F(X, Y, Z=const)* is a natural interval extension for the function *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.

3.2. New approach to contour extraction for AM

So far, IA has been used only in the context in the context of direct slicing based on FRep [7], [50]. However, using the IA approach with the adaptive subdivision algorithm has the drawbacks mentioned above (in 3.1.2); therefore, it is useful to apply an appropriate evolution of IA in FRepCAM, namely, affine arithmetic (AA). This paper describes the use of revised AA for adaptive subdivision of space. Its main purpose is to improve the accuracy of the interval estimation in comparison to IA. Once more, AA is used as an adaptation criterion in the adaptive subdivision of space with quadtrees.

The main ideas of AA can be found in Stolfi and Figueeirdo's introduction [51]. 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

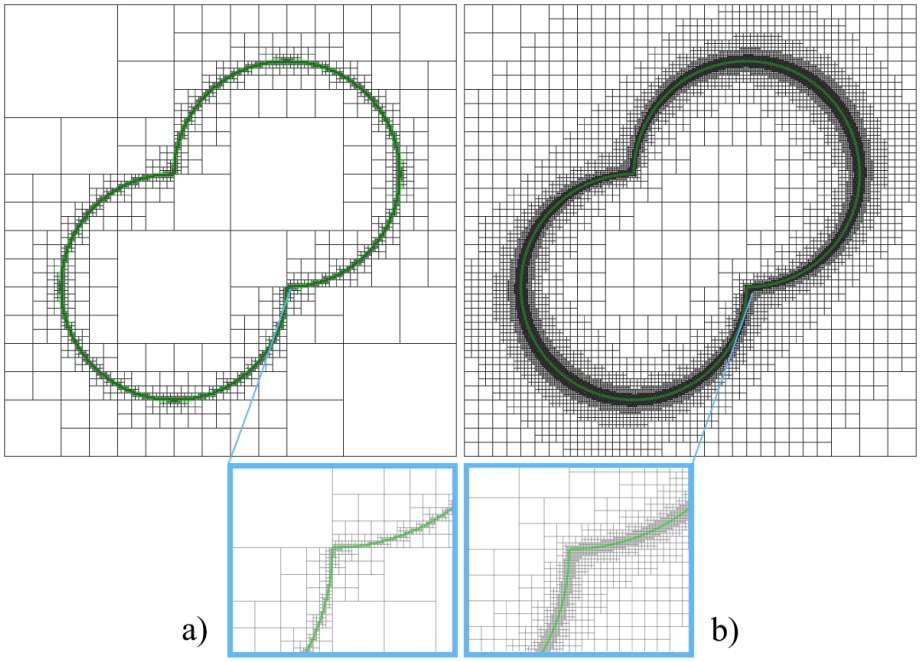


Figure 8: The quadtree construction and the contouring process for one layer of the sphere-union model with an XY resolution of 0.01 mm using the AA criterion (a) and the IA criterion (b). All visited quadrants from all stages of the quadtree construction are shown (black lines). The contour (green line) is built using only cells from the last level of the quadtree. The look-up table of the MS algorithm is used for contour construction.

constructed for each quantity and operation in the defining function. Then, the computations are performed with the affine forms of the defining function. This allows us to obtain an interval estimation of the defining function with more accurate ranges than by using interval arithmetic.

A more detailed description can be found in Appendix A. AA was revised in [52], and several approximations for non-affine operations were improved. Many applicable approximations were collected in [53]. Furthermore, AA was used for the purposes of performing spatial enumeration of implicit surfaces (n being the number of arguments of the defining function, i.e., the dimension [49]).

The revised affine form of a real-valued quantity $\hat{x}$ consists of two parts: the

standard affine part of the length n and the interval part:

$$\hat{x} = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i + e_x[-1,1], e_x \geq 0, \tag{6}$$

where the x_is are finite real numbers and the ε_is 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:

$$\hat{x} = x_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + ... + x_n\varepsilon_n + e_x[-1,1], \tag{7}$$

$$\hat{y} = y_0 + y_1\varepsilon_1 + y_2\varepsilon_2 + ... + y_n\varepsilon_n + e_x[-1,1], \tag{8}$$

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:

$$f(\hat{x},\hat{y}) = (\alpha x_0 + \beta y_0 + \gamma) + \sum_{i=1}^{n}(\alpha x_i + \beta y_i)\varepsilon_i + (|\alpha|e_x + |\beta|e_y)[-1,1], (\alpha,\beta,\gamma \in \mathbb{R}), \tag{9}$$

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:

$$\hat{x} * \hat{y} = (x_0 y_0 + \frac{1}{2}\sum_{i=1}^{n} x_i y_i) + \sum_{i=1}^{n}(x_0 y_i + x_i y_0)\varepsilon_i + e_{xy}[-1, 1], \tag{10}$$

where

$$e_{xy} = e_x e_y + e_y(|x_0| + u) + e_x(|y_0| + v) + uv - \frac{1}{2}\sum_{i=1}^{n}|x_i y_i|, u = \sum_{i=1}^{n}|x_i| \; v = \sum_{i=1}^{n}|y_i|. \tag{11}$$

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, we have to obtain its AA version, which we do 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, we calculate 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, we calculate the defining function value and process the cells in the same way as MS does. The result of this algorithm is a polyline, which we use as a toolpath in our direct fabrication process for the given layer. Algorithm 1 is the resulting algorithm.

The FRepCAM of the model (union of two spheres) using the AA criterion is shown in Figure 8. Here, given a resolution of 0.01 mm, we obtained 29781 cells for IA in 250 ms and 11021 cells for AA in 100 ms. **IA and AA provide us with the same contours, yet AA works 2.5 times faster and reduces the number of explored cells for this model.**

4. Experimental results and a comparison of FRepCAM methods

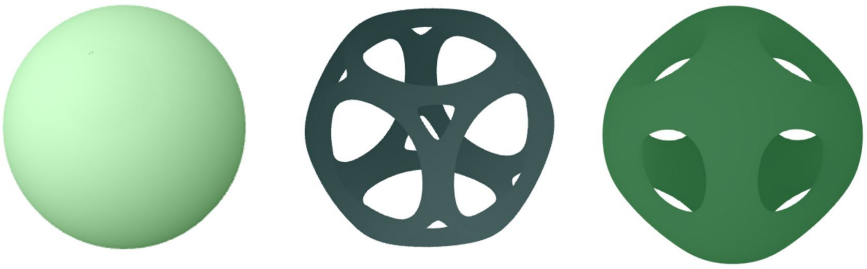
As we have various criteria for FRepCAM, we have compared models of different complexity from a geometric point of view as well as based on the complexity of the defining function. The models were prepared with our approach for direct fabrication, i.e., sliced by using contouring methods, and every contour defined a path for 3D printing hardware. In Table 1, we show different models with respect to the dimensions of their fabricated versions.

Algorithm 1 Contouring with adaptation criteria

BuildQuadTree($x_0, y_0, z_0, BoundingBox$)
Create the topology of the curve using Connected Component Labeling (CCL) algorithm[54]
Calculate the exact values of the implicit curve on the edges of adjacent cells using numerical methods for solving nonlinear equations (bisection method)
procedure BUILDQUADTREE($x, y, z, CellLength$)
    if $CellLength > Precision$ **then**
        ▷ Testing Cell: computation of the interval inside the cell based on the Affine Arithmetic extension of the target function. The interval is checked for zero.
        if AdaptationCriterion($x, y, z, CellLength$) contains 0 **then**
            ▷ Cell division by 4 cells recursively
            $CellLength\ /= 2$
            $BuildQuadTree(x, y, z, CellLength)$
            $BuildQuadTree(x + CellLength, y, z, CellLength)$
            $BuildQuadTree(x, y + CellLength, z, CellLength)$
            $BuildQuadTree(x + CellLength, y + CellLength, z, CellLength)$
        else
            Reject Cell
        end if
    else
        Add Cell as Quadtree Leaf with MS processing
    end if
end procedure ▷ AdaptationCriterion is Affine Arithmetic extension of target function

FRepCAM was 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. We used the template-based Boost library [55] for IA and the authors' implementation of the revised AA [49].

We used algebraic surfaces (Figure 9) along with more complex non-algebraic surfaces (Figure 10) with procedural microstructures and set-theoretic operations, which are based on R-functions [10]. The defining functions of the models can be found in Appendix B.



a) Sphere b) Decocube c) Orthocircle

Figure 9: Algebraic surfaces

To test contouring with various resolutions for the models presented in Table 1, we 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 of AM devices. The resulting timings are shown in Figure 11 and Figure 12.

Figure 11 and Figure 12 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.

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 11). However, for a finer resolution, exhaustive enumeration is no longer efficient.

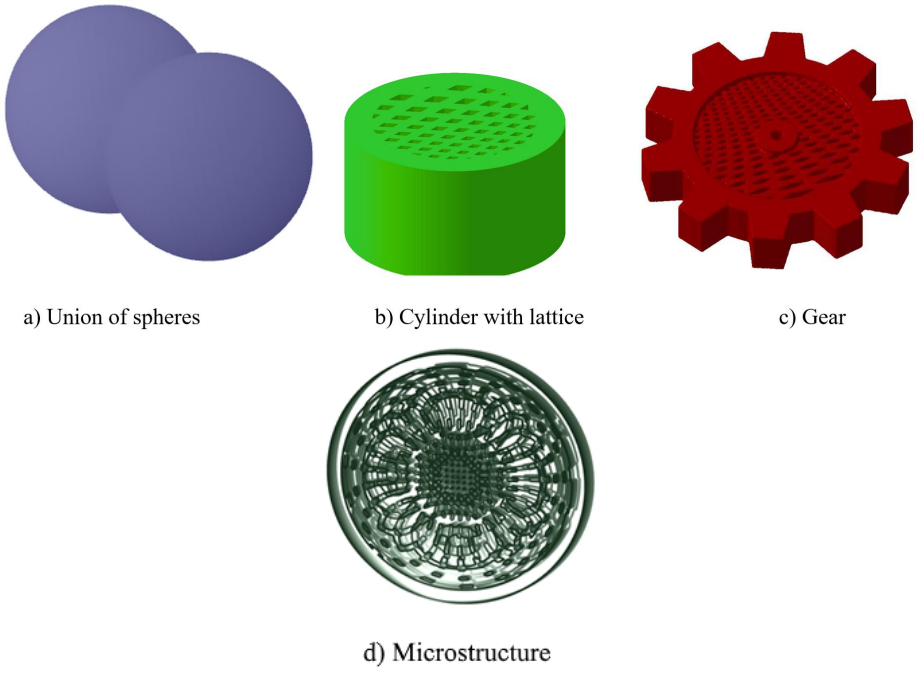


Figure 10: Non-algebraic surfaces

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

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 1: Bounding boxes for contouring

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, which uses a lattice with a specific space mapping (Algorithm 2), and for the gear model. For all considered resolutions, the algorithm with a regular grid works faster. The performance of both adaptive algorithms is the same.

One additional calculation with a 5 μm resolution was performed for the cylinder model (Figure 12, 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 efficiency of adaptive criteria such as IA and AA increases with increasing precision of the calculated curve in one layer. Additionally, we can say that IA and AA work better with algebraic surfaces, especially with quadratic ones. However, the error of overestimation increases when we use transcendental functions or loops for describing complex models. The efficiency of adaptive contouring techniques appears with high XY resolution.

Our recommendations for using algorithms for contouring functionally de-

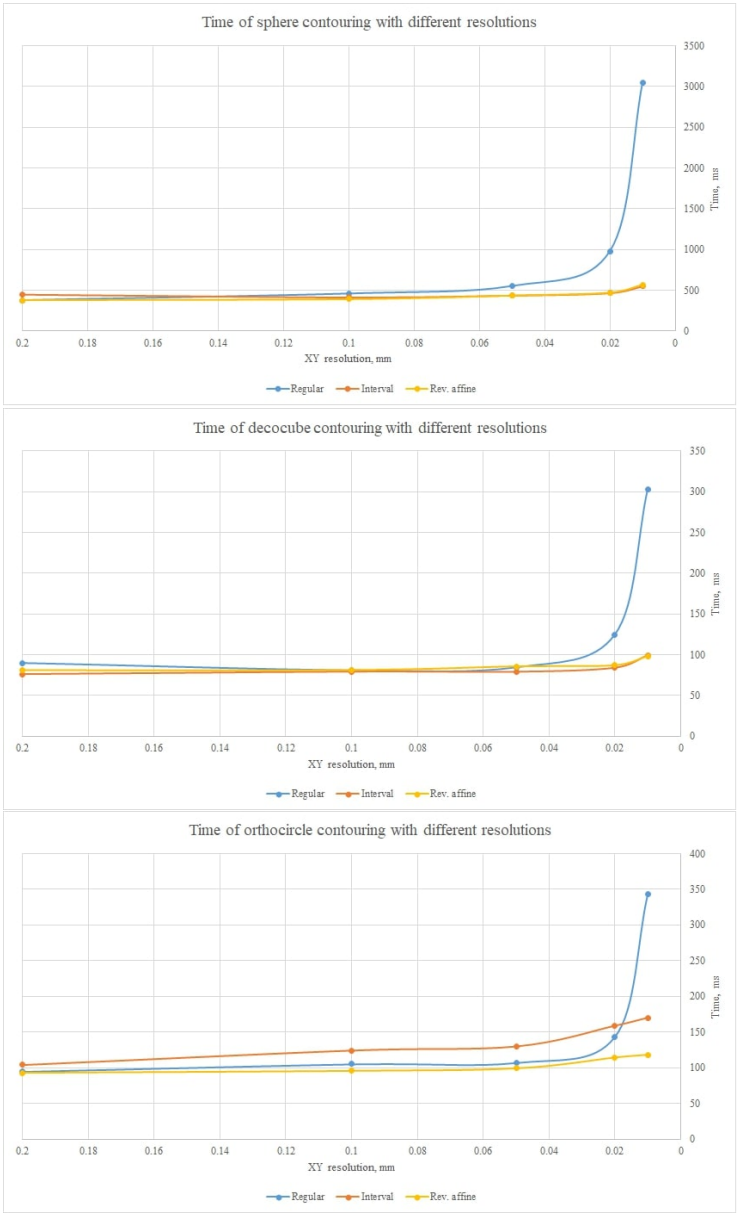
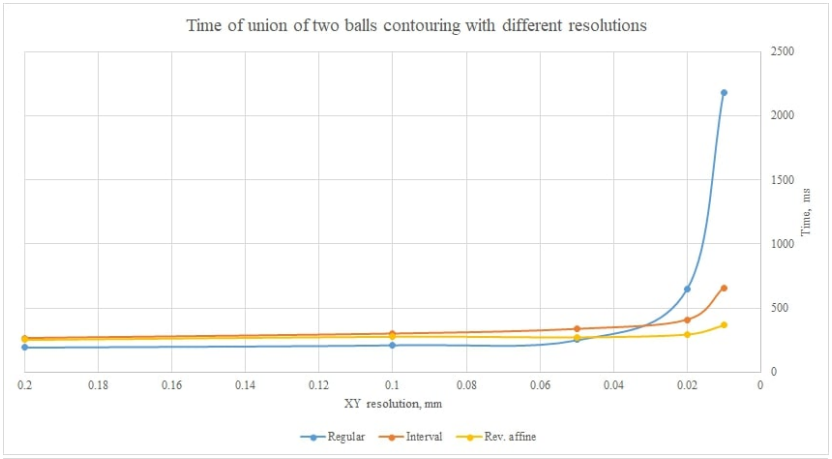
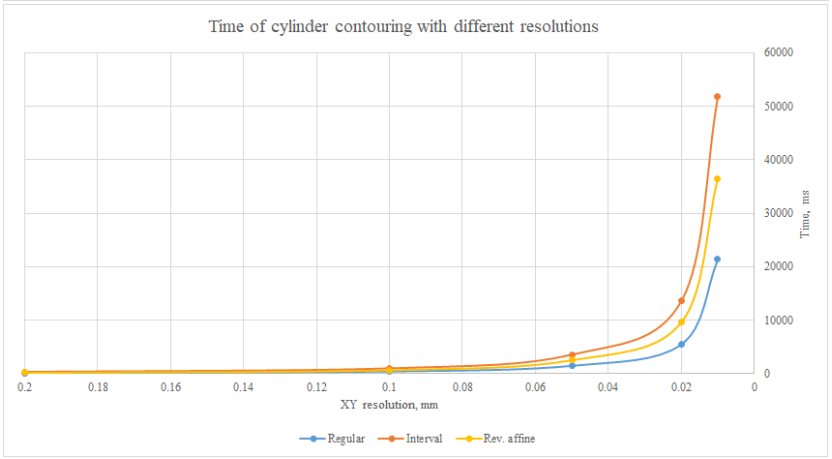
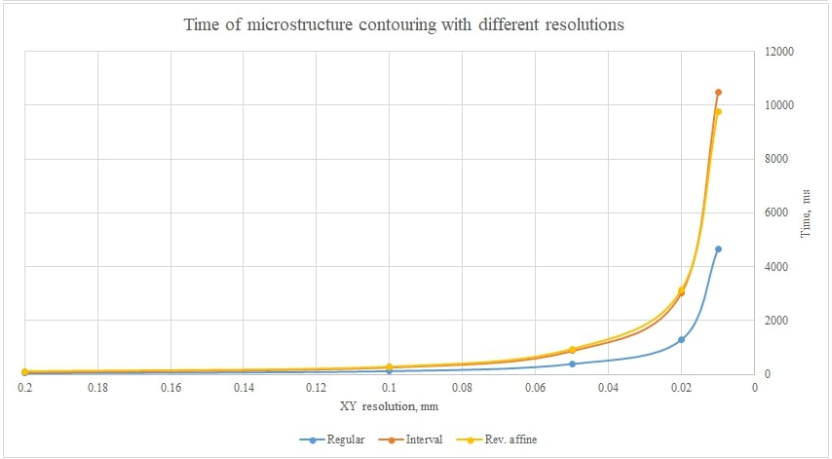


Figure 11: Contouring results for algebraic surfaces

fined 3D objects are summarized in Table 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


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

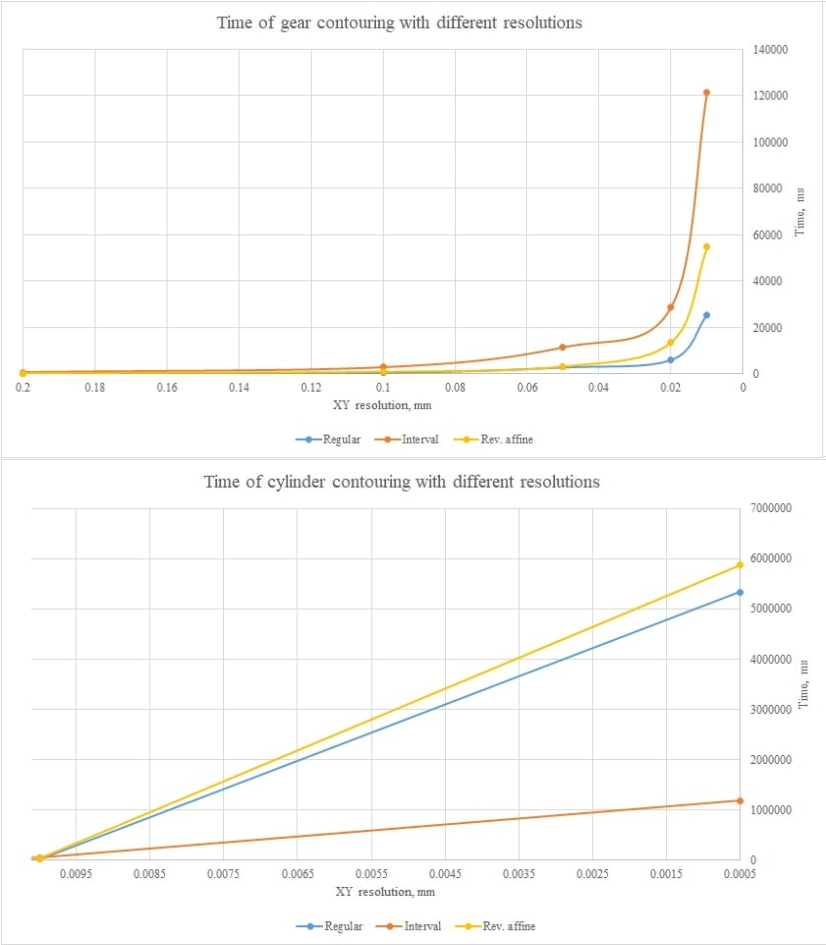


Figure 12: Contouring results for non-algebraic surfaces

complex or smaller model. This means that with increasing model complexity, the use of adaptive methods become reasonable only with finer accuracy. Concerning the choice between adaption criteria, the adaptive subdivision with revised AA is preferred in general.

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. For selective laser sintering and selective laser

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

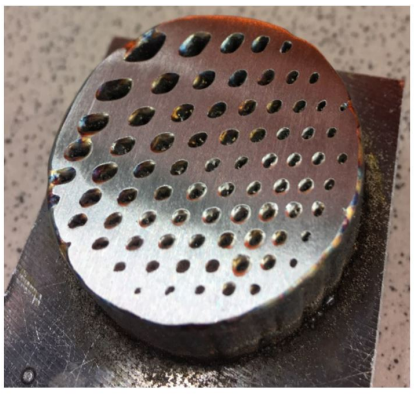
Table 2: Recommended contouring methods for implicit 3D objects

melting (SLS/SLM) and stereolithography (SLA) **with digital light processing (DLP)**, using adaptive algorithms is preferable due to their high resolution near 10 μm.

5. The fabrication of the parts

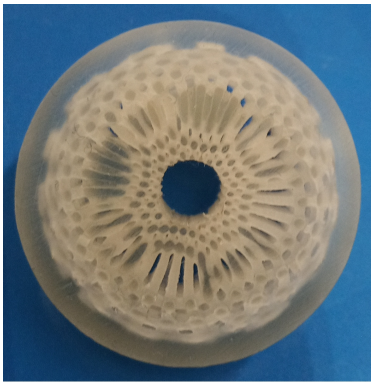
Our test models were fabricated using FDM, DMD and DLP printing processes. The open-source project CuraEngine [56] was used for the generation of supports, infill and GCODE for our FDM and DMD additive manufacturing equipment. In general, this software is applied for the FDM printing process; however, 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 13a. The same software was used for the fabrication of the gear model using the 3D FDM printer Ultimaker S5 (Figure 13b). The 3D DLP

printer Wanhao Duplicator 7 was used for the fabrication of the microstructure model. The 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 13: Printed models: the cylinder with lattice (radius 30 mm, height 60 mm) printed using the 3D DMT printer Insstek MX-1000 (a), the gear (radius 67 mm, height 30 mm) printed using the 3D FDM printer Ultimaker S5 (b), the microstructure (radius 16 mm, height 20 mm) printed using the 3D DLP printer Wanhao Duplicator 7 (c).

6. Conclusions

Direct fabrication, where the toolpath for the hardware is created by omitting the traditional step of conversion into a polygonal representation and subsequently slicing it, allows us to increase the efficiency of using alternative rep-

resentations in the AM process. In our work, we compare different methods for FRepCAM, such as using a regular grid with exhaustive enumeration and recursive methods based on quadtrees with different adaptation criteria. The MS algorithm with the CCL algorithm allows us to build a correct topology of implicit curves. IA and AA can be used as adaptation criteria for improving the algorithm of contour extraction,but the efficiency of their use increases with increasing contouring accuracy. Therefore, algorithms based on exhaustive enumeration might have the same efficiency when high precision is not required. The recommendations given help in choosing the appropriate contouring method according to the 3D printing technology.

Our work represents one of the first steps in CAM systems for fabricating implicit models directly from their FReps without generating a surface mesh. This CAM module will support heterogeneous models on the levels of microstructures and multiple materials. [57]. For microstructures, one can select a resolution arbitrarily. Note that practically any resolution can be selected due to the functional nature of the model. In other models, the resolution is dictated by the given rasterization of the model. For a multi-material model, we detect the number of materials and then apply the proposed algorithm to the defining function of each material separately. In a slice, materials can be deposited appropriately within their obtained boundaries. Note that within their boundaries, materials may remain constant or vary volumetrically. A non-uniform distribution of material inside a boundary can be distinguished during the construction of the quadtree with a certain accuracy level. This is a direction for future research.

Acknowledgements

The work is supported by the Skolkovo Institute of Science and Technology.

Appendix A. Interval and Affine Arithmetic

Appendix 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:

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

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

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

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

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 [45], [58].

Transcendental functions with arithmetic operations are used in 3D modelling. We 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 [34]:

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

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) [59]. 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, we 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.

Appendix 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 $R = X \times Y$ 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.

For the computation of the affine arithmetic form, we must find a good affine approximation f^a for each simple non-affine operation f^*. Different approximation techniques are discussed in [60] 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 [61]:

$$\hat{x}\hat{y} = x_0y_0 + \sum_{i=1}^{n}(x_0y_i + y_0x_i)\varepsilon_i + z_{N+1}\varepsilon_{N+1},$$

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 [61], 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 $s_1 = 2^2 - x^2 - y^2 - z^2, s_1 = 2^2 - (x - 2)^2 - (y - 2)^2 - z^2$.

Cylinder with lattice:

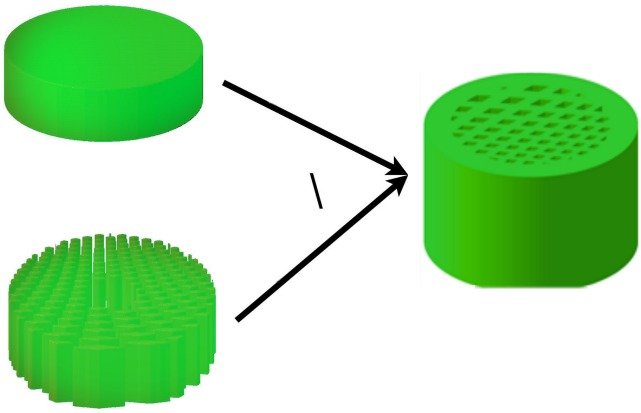


Figure B.1: FRep tree for the cylinder model

Gear:

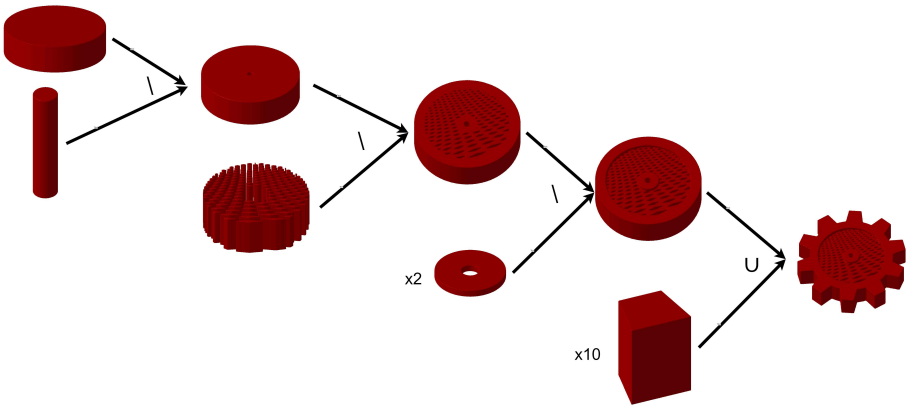


Figure B.2: FRep tree for the gear model

Microstructure:

Algorithm 2 The microstructure model in the HyperFun [62] syntax

```
1: my_model(x[3], a[1]) {
2: R = 16; thickness = 20; l = 0.5;
3: distance = sqrt(x[1]*x[1] + x[2] * x[2]);
4: freqX = 10; freqY = 10; freqZ = 10;
5: sx=sin(freqX*x[1]) - l;
6: sy=sin(freqY*x[2]) - l;
7: sz=sin(freqZ*x[3]) - l;
8: rx=sy & sz;
9: ry=sx & sz;
10: rz=sx & sy;
11: lattice=(rx | ry | rz);
12: BigCylinder = R*R - x[1]*x[1] - x[2]*x[2];
13: SmallCylinder = BigCylinder - thickness;
14: inter = SmallCylinder & lattice;
15: sub = BigCylinder \ SmallCylinder;
16: sub = sub | inter;
17: my_model = sub;
18: }
```

References

[1] M. Burns, Automated fabrication : improving productivity in manufacturing, Englewood Cliffs, N.J. : PTR Prentice Hall, 1993.
URL https://trove.nla.gov.au/work/24193608

[2] A. Aremu, J. Brennan-Craddock, A. Panesar, I. Ashcroft, R. Hague, R. Wildman, C. Tuck, A voxel-based method of constructing and skinning conformal and functionally graded lattice structures suitable for additive manufacturing, Additive Manufacturing 13 (2017) 1–13. doi:https://doi.org/10.1016/j.addma.2016.10.006.

URL http://www.sciencedirect.com/science/article/pii/S2214860416302810

[3] BRITE-EURAM Rapid Prototyping Techniques, Project BE5278. Common Layer Interface (CLI), Tech. rep., BRITE-EURAM (1994).

[4] C. K. Chua, K. F. Leong, C. S. Lim, Rapid Prototyping: Principles and Applications, Singapore: World Scientific Pub Co. Pte. Ltd., 2010, Ch. SLC File Specification, pp. 339–348.

[5] A. Requicha, H. Voelcker, Constructive solid geometry, Tech. rep., College of Engineering & Applied Science. The University of Rochester (Nov. 1977).

[6] I. Gibson, D. Rosen, B. Stucker, Additive Manufacturing Technologies, Springer Science+Business Media, 2015, Ch. Implicit Modeling, pp. 423–426.

[7] Y. Song, Z. Yang, Y. Liu, J. Deng, Function representation based slicer for 3D printing, Computer Aided Geometric Design 62 (2018) 276–293. doi:https://doi.org/10.1016/j.cagd.2018.03.012.
URL http://www.sciencedirect.com/science/article/pii/S0167839618300268

[8] R. Jamieson, H. Hacker, Direct slicing of CAD models for rapid prototyping, Rapid Prototyping Journal 1 (2) (1995) 4–12.

[9] R. Bonnard, J.-Y. Hascoët, P. Mognol, I. Stroud, Step-nc digital thread for additive manufacturing: data model, implementation and validation, International Journal of Computer Integrated Manufacturing 31 (11) (2018) 1141–1160. doi:10.1080/0951192X.2018.1509130.

[10] A. Pasko, V. Adzhiev, A. Sourin, V. Savchenko, Function representation in geometric modeling: concepts, implementation and applications, The Visual Computer 11 (1995) 429–446. doi:10.1007/BF02464333.

[11] V. L. Rvachev, Theory of R-functions and Some Applications, Naukova dumka, Kyiv, 1982.

[12] P.-A. Fayolle, O. Fryazinov, A. Pasko, Rounding, filleting and smoothing of implicit surfaces, Computer-Aided Design and Applications 15 (3) (2018) 399–408. doi:10.1080/16864360.2017.1397890.

[13] O. Fryazinov, T. Vilbrandt, A. Pasko, Multi-scale space-variant frep cellular structures, Comput. Aided Des. 45 (1) (2013) 26–34. doi:10.1016/j.cad.2011.09.007.

[14] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, Reconstruction and representation of 3d objects with radial basis functions, in: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01, ACM, New York, NY, USA, 2001, pp. 67–76. doi:10.1145/383259.383266.

[15] G. Pasko, A. Pasko, T. Vilbrandt, A. Lixandrão Filho, J. Silva, Ascending in space dimensions: Digital crafting of m.c. escher's graphic art, Leonardo 44 (2011) 411–416. doi:10.1162/LEON_a_00241.

[16] R. Rezende, V. Kasyanov, I. Ozolanta, K. Brakke, J. Silva, V. Mironov, Design of vascular tree for organ bioprinting, Computer Aided Chemical Engineering 32 (2013) 151–156. doi:10.1016/B978-0-444-63234-0.50026-9.

[17] J. Brennan-Craddock, D. Brackett, R. Wildman, R. Hague, The design of impact absorbing structures for additive manufacture, Journal of Physics: Conference Series 382 (2012) 1–7. doi:10.1088/1742-6596/382/1/012042.

[18] A. Khoda, B. Koc, Functionally heterogeneous porous scaffold design for tissue engineering, Comput. Aided Des. 45 (11) (2013) 1276–1293. doi:10.1016/j.cad.2013.05.005.

[19] D.-J. Yoo, Advanced projection image generation algorithm for fabrication of a tissue scaffold using volumetric distance field, International Journal of

Precision Engineering and Manufacturing 15 (10) (2014) 2117–2126. doi:10.1007/s12541-014-0571-y.

[20] G. Savio, S. Rosso, R. Meneghello, G. Concheri, Geometric modeling of cellular materials for additive manufacturing in biomedical field: A review, Applied Bionics and Biomechanics 2018 (2018) 1–14. doi:10.1155/2018/1654782.

[21] T. Bodkin, R. Bibb, R. Harris, Towards additive manufacture of next generation prosthetics, assessing emerging cad strategies for improving the existing cad process, International Journal of Rapid Manufacturing 6 (2/3) (2017) 185–196.

[22] B. Zhang, P. Jaiswal, R. Rai, S. Nelaturi, Additive manufacturing of functionally graded material objects: A review, Journal of Computing and Information Science in Engineering 18 (4) (Jul. 2018). doi:10.1115/1.4039683.

[23] G. Loh, E. Pei, D. Harrison, M. Monzón, An overview of functionally graded additive manufacturing, Additive Manufacturing 23 (2018) 34–44. doi:https://doi.org/10.1016/j.addma.2018.06.023.
URL http://www.sciencedirect.com/science/article/pii/S221486041730564X

[24] A. Medeiros e Sa, V. Mello, K. Rodriguez Echavarria, D. Covill, Adaptive voids: Primal and dual adaptive cellular structures for additive manufacturing, Visual Computer 31 (2015) 799–808. doi:10.1007/s00371-015-1109-8.

[25] S. Vongbunyong, S. Kara, Sustainability Through Innovation in Product Life Cycle Design, Shpringer Japan, 2016, Ch. Selective Volume Fusing Method for Cellular Structure Integration, pp. 513–524. doi:10.1007/978-981-10-0471-1_35.

[26] S. Vongbunyong, S. Kara, Rapid generation of uniform cellular structure by using prefabricated unit cells, International Journal of Computer Integrated Manufacturing 30 (8) (2017) 792–804. doi:10.1080/0951192X.2016.1187303.

[27] C. Maple, Geometric design and space planning using the marching squares and marching cube algorithms, in: 2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings, 2003, pp. 90–95. doi:10.1109/GMAG.2003.1219671.

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

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

[30] T. Theußl, T. Möller, E. Gröller, Optimal regular volume sampling, Proceedings Visualization, 2001. VIS '01. (2001) 91–546.

[31] H. A. Carr, T. Theußl, T. Möller, Isosurfaces on optimal regular samples, in: VisSym, 2003.

[32] B. V. Herzen, A. H. Barr, Accurate triangulations of deformed, intersecting surfaces, in: SIGGRAPH '87, 1987.

[33] T. Duff, Interval arithmetic recursive subdivision for implicit functions and constructive solid geometry, in: SIGGRAPH '92, 1992.

[34] N. Stolte, A. Kaufman, Parallel spatial enumeration of implicit surfaces using interval arithmetic for octree generation and its direct visualization, in: Implicit Surfaces'98 Proceedings, 1998, pp. 81–88.
URL https://pdfs.semanticscholar.org/7a21/

36d8222f6c19c5bbc4e30598038bdb75f623.pdf?_ga=2.238635461.496188997.1563888038-540992412.1562229864

[35] G. Wyvill, C. McPheeters, B. Wyvill, Data structure for soft objects, The Visual Computer 2 (2005) 227–234.

[36] H. Rangan, M. Ruhl, D. Saupe, Interactive visualization of implicit surfaces with singularities, Comput. Graph. Forum 16 (1997) 295–306.

[37] K. Levinski, A. Sourin, Interactive polygonisation for function-based shape modelling, 2002.

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

[39] J. Bloomenthal, B. Wyvill, Introduction to implicit surfaces, 1997.

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

[41] B. R. de Araújo, D. S. Lopes, P. Jepp, J. A. Jorge, B. Wyvill, A survey on implicit surface polygonization, ACM Comput. Surv. 47 (4) (2015) 60:1–60:39. doi:10.1145/2732197.

[42] G. Wyvill, C. McPheeters, B. Wyvill, Data structure for soft objects, The Visual Computer (1986) 227–234.

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

[44] D. Kalra, A. H. Barr, Guaranteed ray intersections with implicit surfaces, in: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '89, ACM, New York, NY, USA, 1989, pp. 297–306. doi:10.1145/74333.74364.

[45] R. Moore, Interval Analysis, Englewood Cliff, New Jersey: Prentice-Hall., 1966.

[46] T. Sunaga, Theory of an interval algebra and its application to numerical analysis, Japan Journal of Industrial and Applied Mathematics 26 (2009) 125–143.

[47] W. Warmus, Calculus of approximations, Bull. Acad. Polon. Sci. Cl. III. 4 (1956) 253–259.

[48] B. Katja, Implicit linear interval estimations, in: Proceedings of the 18th Spring Conference on Computer Graphics, SCCG '02, ACM, New York, NY, USA, 2002, pp. 123–132. doi:10.1145/584458.584479.

[49] O. Fryazinov, A. Pasko, P. Comninos, Fast reliable interrogation of procedurally defined implicit surfaces using extended revised affine arithmetic, Comput. Graph. 34 (6) (2010) 708–718. doi:10.1016/j.cag.2010.07.003.

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

[51] J. Stolfi, L. de Figueiredo, An Introduction to Affine Arithmetic, TEMA Tend. Mat. Apl. Comput. 4 (3) (2003) 297–312. doi:https://doi.org/10.5540/tema.2003.04.03.0297.
URL https://tema.sbmac.org.br/tema/article/view/352

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

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

[54] L. G. Shapiro, Connected component labeling and adjacency graph construction, 1996.

[55] G. Melquiond, S. Pion, H. Brönnimann, Boost library. interval arithmetic library, last accessed 19 February 2019 (2006).
URL https://www.boost.org/doc/libs/1_66_0/libs/numeric/interval/doc/interval.htm

[56] Ultimaker, Curaengine, last accessed 19 February 2019 (2013).
URL https://github.com/Ultimaker/CuraEngine

[57] A. A. Pasko, V. Adzhiev, B. Schmitt, C. Schlick, Constructive hypervolume modeling, Graphical Models 63 (2001) 413–442.

[58] J. Snyder, Interval analysis for computer graphics, in: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '92, ACM, New York, NY, USA, 1992, pp. 121–130. doi:10.1145/133994.134024.

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

[60] d. F. L. H. Stolfi, J., Self-Validated Numerical Methods and Applications, IMPA, 1997.

[61] L. de Figueiredo, J. Stolfi, Affine arithmetic: Concepts and applications, Numerical Algorithms 37 (2004) 147–158. doi:10.1023/B:NUMA.0000049462.70970.b6.

[62] C. Richard, V. Adzhiev, A. Pasko, Y. Goto, T. Kunii, Web-based shape modeling with hyperfun, IEEE computer graphics and applications 25 (2005) 60–9. doi:10.1109/MCG.2005.49.