Communications of JSSAC (2016)
Vol. 2, pp. 33 – 42

Practice of Drawing Graphs of Implicit Functions of Three Variables

Noriko Hyodo
Salesian Polytechnic

Yuji Kondoh
National Institute of Technology, Kagawa College

Hirokazu Murao
The University of Electro-Communications

Tomokatsu Saito
AlphaOmega Inc.

Tadashi Takahashi
Konan University

(Received 31/Oct/2014 Accepted 16/Feb/2015)

Abstract

We have long been working on developing algorithms for drawing graphs of implicit functions. In this paper, we investigate methods for trivariate cases as a simple extension of the existing methods for bivariate cases. The basic strategy of our methods consists of division of the 3D space in the display area into a contiguous sequence of tiny cubes, and the check for every cube of the existence of the solutions to the polynomial equation defining an implicit function. The method itself is characterized by the exact treatment with mathematical preciseness of numeric calculation and formula manipulation provided by computer algebra system. The underlying mathematics used in our most precise algorithm for bivariate cases cannot be extended direct to trivariate cases, and there remains a small chance even for our current most rigorous algorithm to miss the detection of a closed curve of solutions if it is isolated inside a cube. We explain how we can simply extend our method to trivariate cases, and investigate the situations of the solution graphs that each algorithm may miss. We also show some sample results obtained by our several experimental implementations.

1 Introduction

There have been developed many mathematical software systems which support the functionality of graph drawing. Visualization is useful for roughly grasping the behavior of mathematical functions. It is important with visualization to present the overall behavior exactly and also not to lose critical nature at some distinct points. Despite this, how to establish an accurate drawing method or the preciseness of drawing methods are rarely discussed. Drawing the figure of the real zeros of the equation of a multivariate polynomial, in other words, the graph of a real algebraic function, is an old problem hard to solve[1]. In general, the problem to obtain the real solution curves of a polynomial exactly still be very difficult, because they may have isolated points or curves. Actually, not a few existing software systems may fail without any notice. In this paper, we treat polynomial equations with rational coefficients, and describe a method to plot the solution curves precisely.

The computer algebra system Risa/Asir[2] equips with a function, called *ifplot*, for plotting the curves of implicit functions defined by bivariate polynomials. The method of plotting is based on a principle for a graph to be mathematically precise, and on this principle, an algorithm is developed for obtaining the solution curves. The effectiveness of the proposed principle and the practicality of the algorithms has been demonstrated by examples. More concretely, the method makes, internally, use of an evaluation function of a square region to tell whether it contains any solution(s) of an implicit function. We call this evaluation function *character* function (or *character* for short). Every region covering the display area is checked by a character function. Character function can be implemented in various ways, depending on the requirement for efficiency and mathematical correctness in practice. The ultimate condition expected for character is to guarantee the existence or non-existence of solutions. As described in [4] and implemented with `ifplot`, such an ultimate character can be realized for 2D cases by making use of Gröbner basis calculation. This method cannot be directly extended to 3D cases.

Nowadays, various types of devices for 3D plotting are available, such as 3D display terminals and 3D printers, and they are becoming very common in recent years. Furthermore, processors widely used have obtained excessive high-performance, and it is expected that a huge amount of calculations required for 3D graphics can be processed with reasonable efficiency by utilizing the performance. With this recognition in mind, we shall treat implicit functions of three variables in our plotting context with Risa/Asir, and we investigate our method toward the 3D extension in this paper.

2 Fundamentals

2.1 Basic Concepts and Principle

Drawing the graph of mathematical functions is commonly recognized as an easy problem, despite the difficulty in mathematical precise treatment. Every existing display device of any kind consists of a large amount of "spots", each of which has finite area or volume determined by the resolution. A curve or a graph is usually drawn, regardless of connected or non-connected, as a set of spots, thus, having some area or volume, although a point or a curve cannot have a width or thickness in mathematical senses. Spot is often termed as *pixel* in 2D cases and *voxel* in 3D cases. Then, in what sense and to what extent the drawn figure is precise? It is difficult to give an answer to this question in a mathematical well-defined manner. Notice that a graph shows only a sketch of the true mathematical curve. By employing algebraic computation, however, we can give a mathematically definite meaning of the drawn figure of the solution curves. In [5, 4], the authors described, starting from the definitions of mathematical basic notions, the abstract meaning of plotting, and how we can improve the mathematical preciseness in the computing methods actually used in practice, in the context of plotting in the 2D space. As a preparation for the 3D-extension, we give a brief review of the previous works by quoting some of the descriptions in [4] in the following.

Let D be a connected compact subset of $\mathbb{R}^n$, and let $f : D \to \mathbb{R}$ be a function defined on D and continuous. As in [5], we propose the following principle for plotting the solutions to the equation $f = 0$. We shall use a *cell* as a terminology of the generalization of spot. The required properties for cells will be explained later.

Plotting principle 1. A *cell* has an n-dimensional volume.

2. A cell is *plotted* if it has a point in $\mathbb{R}^n$ in common with the solutions, and any non-plotted cell has no common point with the solutions.

Here, *plotted* means being displayed with a foreground color on the displaying device.

A plotting algorithm that satisfies the above principle is not easy to develop in general. The difficulty lies in constructing effective procedure to decide whether a solution exists or not in a given area. Required functionalities for an algorithms are outlined below.

- division of the domain of plotting area into a family of small sets of points, called *cells*.
- method to determine whether the given equation has isolated singular points, and if it does, the location of the cells in which the isolated singular points exist.
- method to determine whether a specified cell has any solution on its boundary.
- method to determine whether a specified cell contains any solutions which have no intersections on the boundary, i.e., entirely contained closed curve/surface.

2.2 Mathematical Description of Plotting

We describe the meaning of plotting with mathematical preciseness. On the above principle, plotting the solution of $f = 0$ is to decide for every cell defined on D whether or not it intersects the solution set $\{(x_1, x_2, \ldots, x_n) \in D \mid f(x_1, x_2, \ldots, x_n) = 0\}$. Let $\{C_j \mid j = 1, ..., m\}$ be a family of m subsets of D.

Definition 1 *We call $\{C_j \mid j = 1, ..., m\}$ a family of cells, or resolution, defined on D if every C_j is a connected closed subset of D, $D = \bigcup_{i=j}^{m} C_j$, and ${C_j}^{\iota} \bigcap {C_k}^{\iota} = \emptyset$ for $j \neq k$ where ${C_j}^{\iota}$ and ${C_k}^{\iota}$ denote sets of all interior points of C_j and C_k respectively.*

Our main concern in plotting the zeros of a function is to compute the following function called a *character*.

Definition 2 *We call a function $\chi : C_j \to \{0, 1\}$ a character of f with resolution $\{C_j\}$ defined on D if $\chi(C_j) = 0$ implies $f(x_1, x_2, \ldots, x_n) \neq 0$ for every point $(x_1, x_2, \ldots, x_n)$ in C_j.*

A character function χ guarantees that if $\chi(C_j) = 0$ then the given function f never vanish all over the cell C_j. It does not guarantee the existence of zeros of f in the cell C_j when $\chi(C_j) = 1$, however. The condition for character is not sufficiently tight to be used in practice for plotting the zeros of a function, in general. We propose a strong property of character as follows.

Definition 3 *A character χ of f with resolution $\{C_j\}$ on D is faithful if $\chi(C_j) = 1$ implies that there exists a point $(x_1, x_2, \ldots, x_n)$ in C_j such that $f(x_1, x_2, \ldots, x_n) = 0$.*

The faithful character provides desired functionality for exact plotting. For general bivariate functions, there are no known algorithms to compute faithful character, except for the one which applies to bivariate polynomials with rational coefficients [3].

We consider how we can implement a character function concretely. Hereinafter, we limit our concern to such cases that the family $\{C_j\}$ is composed of rectangular grid points, and we let C_k be $\{(x_1, ..., x_n) \mid a_{k,l} \leq x_l \leq b_{k,l},\ 1 \leq l \leq n,\ a_{k,l} \in \mathbb{Q}, b_{k,l} \in \mathbb{Q}\}$. Let $I_{k,l}$ denote an interval $[a_{k,l}, b_{k,l}]$.

Interval character. The function that checks if zero is contained in the interval value $f(I_{k,1},...,I_{k,n})$ satisfies the condition for character, but cannot be faithful in most cases. This function is called an *interval character*.

In general, it is quite difficult to give a concrete computing method for a character function, even without requiring the faithfulness. We let ourselves get pragmatic. We focus on the detection of zeros of a given polynomial, and introduce a notion of functions, called *weak character*, which are not necessarily character (in the sense of Definition 2) but satisfy the property of Definition 3. Also, in the following, we use the same term "character" referring to the functions used for the determination of cell plotting.

2.3 Plotting in 2D-space

We describe our basic ideas used for 2D cases, i.e., for plotting zeros of a bivariate polynomial $f(x,y)$. Without loss of generality, we assume f is square-free for simplicity.

According to its definition, for a specified cell, character must detect and must not miss the existence of any solution to the equation $f(x,y) = 0$ in the cell. Consider the case when a certain cell contains the solution of $f = 0$, and consider how to construct a function to determine the existence of any solutions. We consider what could likely to happen with the cells containing the zeros, and consider how it can be detected. In most cases, the curve would intersect with any of the edges of the cell, and the existence of any solution on a line segment can be easily checked, e.g., using the signs of the both endpoints or Sturm's theorem. These observations lead to the development of a series of character functions actually used in our implementation of `ifplot` for 2D cases as follows.

Sign (weak) character. Consider a certain cell C_k containing the solutions to $f = 0$. Then, it is likely that the signs of f are not all equal at four corners of C_k, i.e., $(a_{k,1}, a_{k,2})$, $(b_{k,1}, a_{k,2})$, $(a_{k,1}, b_{k,2})$ and $(b_{k,1}, b_{k,2})$. Our simplest character function, called sign (weak) character, computes and checks those signs.
Even if the curve $f = 0$ has any common point with any of the edges, this character may have a chance to miss their existence as in the case that the curve intersects with one edge an even number of times by taking their multiplicities into account. This type of omission can be avoided simply by computing Sturm's sequence.

Boundary (weak) character. If a certain cell C_k contains the zeros of f, then it is very likely that f has its zeros on any of the four edges. Boundary (weak) character is designed to detect the existence of the zeros on the edges of a cell. According to Sturm's theorem, we can determine the number of zeros of a univariate polynomial existing in a specified interval, and we shall use this theorem to detect the existence of zeros. More concretely, for each cell C_k, we check whether at least one of univariate polynomials $f(a_{k,1}, y)$, $f(b_{k,1}, y)$, $f(x, a_{k,2})$ and $f(x, b_{k,2})$ have zeros in the intervals $I_{k,1}$ for x and $I_{k,2}$ for y. In practice, we may recursively apply the bisection method and the detection by using Sturm's theorem to each of full line segments of the grid in the display area, until the interval of the bisected segments gets smaller than the resolution or non-existence of zeros in the segment is confirmed, as in [6].

Both of the above two characters will fail to detect the existence of zeros in a cell if all the zeros are of singular points or closed curves completely isolated inside the cell.

Implementation approach to a faithful character. We assume that f is not only square-free but irreducible. Singular points, if exist, can be obtained as solutions to the zero-dimensional system of polynomial equations $f = \partial f/\partial x = \partial f/\partial y = 0$. Also, on every closed curve, there must exist finitely many points that satisfy the system of equations $\partial f/\partial x = 0$ or $\partial f/\partial y = 0$. Therefore, we can determine the locations of any cells containing isolated zeros by computing the solutions via the Gröbner basis of the system with sufficient accuracy that makes the precision of the locations smaller than the resolution.

This way of algebraic treatment plus the previous method for the detection on boundary can establish a faithful character.

3 Extension for Plotting in 3D-space

We now consider the case with three variables, i.e., to plot zeros of trivariate implicit functions $f(x, y, z) = 0$, in 3D space. Most of the arguments and the algorithms for 3D cases parallel those of 2D cases. Our overall strategies and the algorithms derived from the intermediate value theorem and Sturm's theorem are made so simple as the mathematical correctness and the accuracy of numeric calculation may rely on computer algebraic computation, and can be easily expanded to 3D cases. However, there arises a difficult problem in 3D cases, if there exists an isolated closed surface inside a voxel and we need to identify the precise location of the voxel. The rest of the section is devoted to investigating the algorithms using various characters, towards the 3D-extension, to some details from the practical point of view.

3.1 Character Functions for Graphs in 3D-space

The basic strategy of our algorithms is to check the existence of solutions against all cells or voxels in the drawing area, using character function, and the algorithms differ in how character determines the (non-)existence of solutions. We assume that all numeric calculations are done exactly or with sufficient accuracy.

3.2 Sign (Weak) Character

The simplest algorithm uses sign (weak) character. Based on the intermediate value theorem, sign (weak) character determines the existence of zero from the signs of $f(x, y, z)$ at the corners of a voxel. The procedure of the algorithm using sign (weak) character consists of the following steps.

1. fix grids defining a set of voxels and fix the coordinates of all grid points in a display area
2. evaluate $f(x, y, z)$ at all the grid points, and determine the zeroness or the signs of the values
3. if the values at all corner points of one voxel are non-zero and have the same sign, the voxel is regarded containing no zeros.

   This algorithm has such merits as

- required calculations can be done efficiently, and especially, can be performed in parallel, and
- it can be applied not only to algebraic equations but any continuous functions.

Notice that the algorithm is valid if the function to plot behaves gently, i.e., the value of function changes slowly in the range of the coordinates of a single voxel. On the other hand, the algorithm may miss the existence of zeros in such cases as the solution exists as an isolated singular point in a voxel, the solution curve/surface in a voxel is completely isolated or contained in a voxel, the solution curve/surface in a voxel is closed and its outward extension to the neighboring voxel, if any, circumvent the corner points, and so on.

3.3 Boundary Character

Boundary character is designed to detect the existence of solutions on the boundary edge or face. In 2D cases, Sturm's theorem can be used to detect the solution on the edge, a boundary of cell [3]. The boundary of each voxel consists of six faces of a cube. If a voxel contains any solution, the solution surface/curve is very likely to intersect with at least one of the six faces of the voxel cube. Every voxel face is a grid square of some grid plane. We consider the intersection curve(s)/point(s) of the surface/curve and each grid plane. On each grid plane, we consider the polynomial $f(x, y, z)$, i.e., the bivariate polynomial $f(x, y, z)$ obtained by the substitution corresponding to the plane, and we determine a set of squares of the plane containing its zeros. For this determination, we can use our method for 2D cases, and especially for completeness, we have only to use faithful character algorithm. Notice that faithful character for 2D cases can determine the existence or non-existence of solutions in a cell exactly. Therefore, boundary character for 3D cases can never miss the existence of a solution as long as the surface/curve has a point in common with any of the boundaries of voxel. The only case that this algorithm may miss is those when the graph of the solution is completely isolated inside a voxel. Treatment for those cases will be considered next.

3.4 Faithful Character

The role of character function is ideally the exact determination of the existence of solutions in a specified one of regions, cells or voxels, equally divided by grids. We want detect even those 3D cases when the curve/surface is isolated completely inside a single voxel. The isolated curve/surface must be closed, and as in 2D cases, the condition satisfied other than $f = 0$ differs depending on the shape of the curve/surface. If the isolated curve/surface is a single point, it is singular at the point and therefore, the following must be satisfied at the point:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = \frac{\partial f}{\partial z} = 0. \tag{1}$$

In the case of closed surface, there always exists a point at which the tangent is parallel to the axis of x, y or z, say x afterwards, and then, the partial derivative of f in that direction must be equal to 0 at the point. Usually, it is expected that there exist only a finite number of such points, in which

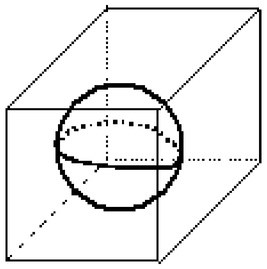

Fig. 1: Example: a closed surface completely isolated inside a voxel.

case the ideal $< f, \partial f/\partial x >$ is zero-dimensional and the solutions of the polynomial system of the ideal determine the voxel position containing the surface. The above discussion can be summarized as follows.

1. Taking the direction of x-axis as one direction, we compute the Gröbner basis of polynomials f and $\partial f/\partial x$.

2. If the ideal turned out to be zero-dimensional, solve the polynomial system to obtain its zeros.

This process can always detect a single point of the graph component isolated inside a single voxel, and is very likely to detect a closed surface inside a single voxel.

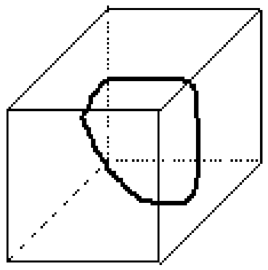

Fig. 2: Example difficult to detect: a closed curve completely isolated inside a voxel.

Very special is the case of closed curves, where the sign of f does not change around the curve, and therefore the condition (1) is satisfied on every point of the curve. This means that the ideal obtained by computing the Gröbner basis of f and some of the partial derivatives of f w.r.t. x, y and z is of positive dimension, and the method for detection mentioned above cannot be used. As a very simple example of this, we give the following equation

$$f(x,y,z) \quad = \quad (x^2+y^2+z^2-1)^2+x^2=0$$

representing the unit circle of the intersection of a sphere and a plane, and we assume the voxel has such a wide range that the circle is completely contained in a single voxel. The Gröbner basis will give a set of polynomials of the plane and the circle. In such rare and simple cases of closed curves, the location of the voxel containing a closed curve can be easily determined by some means, however, in general, more advanced algorithm as cylindrical algebraic decomposition(CAD) will be necessary. Further investigation is left for future study.

4 Empirical Study

As described in the previous section, it is difficult to guarantee the mathematical preciseness in plotting trivariate implicit functions, and even with our most precise algorithm using faithful character, there remains a chance to miss a closed curve complete isolated inside a single voxel. However, from a practical point of view, there may be a chance that such a tiny graphical component need not be drawn because it is not visible, and if being very casual is allowed, we may say that such a tiny component does not affect on the total appearance in most cases. We therefore, being very pragmatic, started to implement the 3D-extension of our algorithm using Risa/Asir for empirical study. In what follows, we report a part of our study, and show some examples.

Using Risa/Asir language The user language of Risa/Asir is equipped with a function to plot a given set of coordinate data, in order to facilitate experimentation of drawing method. Our first attempt was done by using this facility, and the algorithm of boundary character was implemented. Figures 3 and 4 are the sample output of this implementation. The defining polynomials used are as follows:

$$f(x,y,z) \quad = \quad (x^2+y^2+z^2)^2-10(x^2+y^2)+6z^2-10=0 \tag{2}$$

$$f(x,y,z) \quad = \quad (x^2+y^2+z^2-1)^2+(x-1/32)^2=0 \tag{3}$$

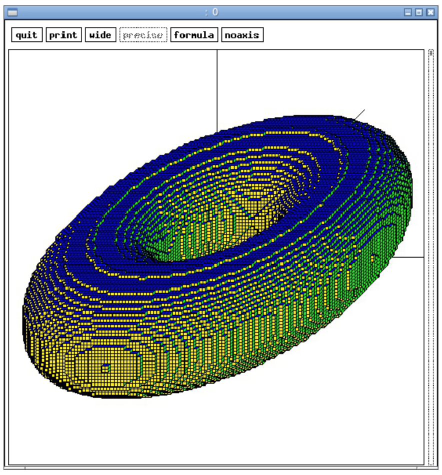

Fig. 3: Drawing example of Eq. (2)

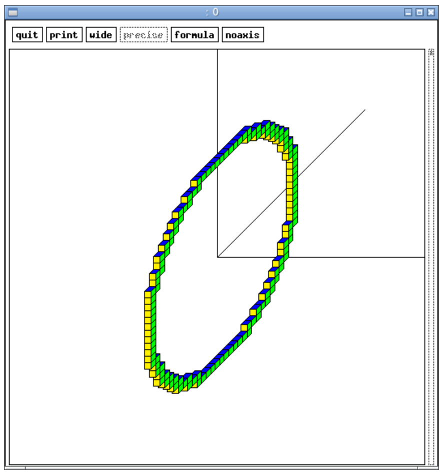

Fig. 4: Drawing example of Eq. (3)

Fig.3 represents the graph of Eq. (2) in $128 \times 128 \times 128$ grid, and Fig.4 does the graph of Eq. (3) in $64 \times 64 \times 64$ grid. These examples indicate that boundary character is sufficiently useful for this level of preciseness.

Using OpenGL for Drawing Graphical capability provided by the user language of Risa/Asir is quite limited; even standard capabilities such as for scaling or for rotation are not available. Also, the drawing speed is as fast as being tolerable for one shot drawing, but is quite slow by today's graphics standard. To remedy these defects, we developed a new drawing function which makes use of OpenGL. OpenGL provides rich functions for graphics manipulation, which will ease the development of standard graphics capabilities. We still use the implementation in Risa/Asir user language for calculations with character functions, and we simply replace the drawing part with our new implementation using OpenGL. Two types of character functions are implemented, boundary character and sign (weak) character. We use the boundary character in the following examples.

Figures 5, 6 and 7 are the examples drawn by our new implementation in $128 \times 128 \times 128$ grid, where the implicit functions used are defined respectively by the following equations:

$$f(x,y,z) = 16(x^2+y^2+z^2)^2 - 40(x^2+y^2) + 24z^2 - 9 = 0, \tag{4}$$
$$f(x,y,z) = (x^2+y^2+2z^2-1)^3 - x^2y^3 = 0, \tag{5}$$
$$f(x,y,z) = (x^2+y^2+2z^2-1)^2 + (x-1/32)^2 = 0. \tag{6}$$

With our new implementation, remarkable speedup has been attained, and also, we recognized again that boundary character is sufficient for practical use. Additionally, we should mention that the use of OpenGL, rather than the direct use of X as in the current implementation in Risa/Asir, makes it easier to port to a wide variety of platforms and to develop various graphical capabilities. The experience and the result of our empirical study strongly support that the hard problem of drawing 3D graphs of trivariate implicit functions can be treated with sufficient preciseness and reasonable amount of computing time, and that our 3D-extended method is useful in practice.

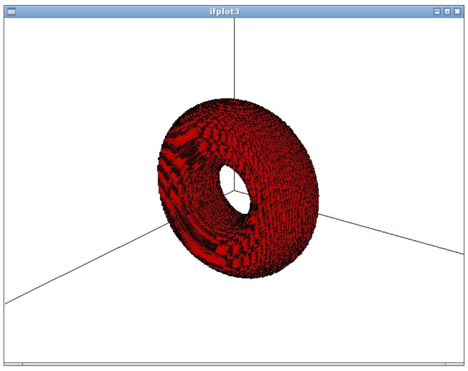

Fig. 5: Drawing example of Eq. (4) by OpenGL

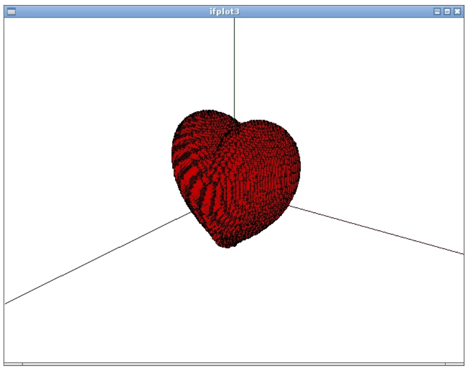

Fig. 6: Drawing example of Eq. (5) by OpenGL

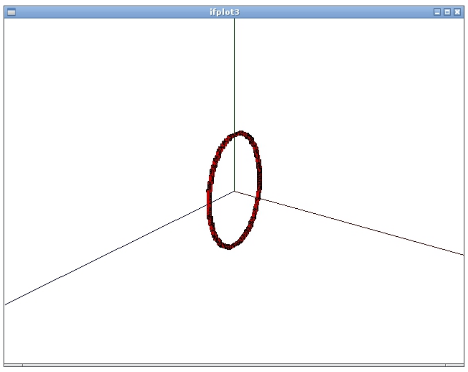

Fig. 7: Drawing example of Eq. (6) by OpenGL

5 Conclusion

In this paper, we investigated how we can draw 3D graphs of trivariate implicit functions exactly, and reported our empirical study with some experimental implementation. The methods used and implemented are the straightforward 3D-extension of the existing methods for 2D cases, each of which uses a different type of character function. Our voxel-based algorithm checks every voxel in the display area whether it contains the zero of the implicit function by character function. For each type of character function, we explained what situation of point/curve/surface may have a chance to be not detected. Even faithful character, our sharpest character, may miss a closed curve isolated inside a voxel, in which case the existence of such curve itself can be detected but its (voxel) location cannot be identified by our simple algorithm.

Finally, we would like to point out the similarity and the affinity of our voxel-based algorithm with 3D printers. The use of voxel as a smallest of plotting will be convenient if 3D printers are targeted as an output device, because a pixel on printing device can be treated as one voxel. Some typical type of 3D printers construct printed object by stacking layers of material, and the processing structure of repetition of the voxel-based algorithm is same as the layer-by-layer printing process and can be applied to the process. Notice that actual printing has such a problem as how to add supports for points or parts of object floating in the air.

As explained before, our current algorithm is still incomplete. Investigation of appropriate methods for detecting a closed curve isolated inside a single voxel and development of mathematically correct and complete algorithm are left for further study.

References

[1] R. Fateman. Honest plotting, global extrema, and interval arithmetic. In Wang [7], pages 216–223.

[2] M. Noro and T. Takeshima. Risa/Asir — a computer algebra system. In Wang [7], pages 387–396.

[3] T. Saito. An extension of Sturm's theorem to two dimensions. *Proceedings of the Japan Academy, Ser. A Mathematical Sciences*, 73(1):18–19, 1997.

[4] T. Saito. *Displaying Zeros of Mathematical Equations*. PhD thesis, 2000. (in Japanese).

[5] T. Saito, Y. Kondoh, Y. Miyoshi, and T. Takeshima. Displaying real solution of mathematical equations. *Journal of JSSAC*, 6(2):2–21, 1998. (in Japanese).

[6] T. Saito, T. Takeshima, and T. Hilano. *Practice and Application of Gröbner Basis Computation*. University of Tokyo Press, 2003. (in Japanese).

[7] P. S. Wang, editor. *Proceedings of ISSAC '92*, Berkeley, CA, July 27–29 1992.