A fast algorithm for the adaptive discretization of 3D parametric curves

Jianming Zhang, Chuanming Ju and Baotao Chi
State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University, Changsha, China

Received 3 June 2019
Revised 15 September 2019
9 October 2019
Accepted 11 November 2019

Abstract

Purpose – The purpose of this paper is to present a fast algorithm for the adaptive discretization of three-dimensional parametric curves.

Design/methodology/approach – The proposed algorithm computes the parametric increments of all segments to obtain the parametric coordinates of all discrete nodes. This process is recursively applied until the optimal discretization of curves is obtained. The parametric increment of a segment is inversely proportional to the number of sub-segments, which can be subdivided, and the sum of parametric increments of all segments is constant. Thus, a new expression for parametric increment of a segment can be obtained. In addition, the number of sub-segments, which a segment can be subdivided is calculated approximately, thus avoiding Gaussian integration.

Findings – The proposed method can use less CPU time to perform the optimal discretization of three-dimensional curves. The results of curves discretization can also meet requirements for mesh generation used in the preprocessing of numerical simulation.

Originality/value – Several numerical examples presented have verified the robustness and efficiency of the proposed algorithm. Compared with the conventional algorithm, the more complex the model, the more time the algorithm saves in the process of curve discretization.

Keywords Computer-aided engineering, Meshing, Curve discretization, Iterative algorithm, Mesh, Computer graphics

Paper type Research paper

1. Introduction

The meshing of three-dimensional models is of the most important in the areas of computer graphics and computer-aided engineering, such as structural analysis by finite element method (FEM), boundary element method (BEM) and among others. The meshing quality has a strong influence on the accuracy, the stability and even computational efficiency of numerical solution (Cuilliere, 1997; Cuillière, 1998; Wu and Wang, 2005; Laug and Borouchaki, 2004; Bournival *et al.*, 2010; Cunha *et al.*, 1997; Thompson, 1999). Reasonable size and density distribution are the essential components of eligible mesh generation. Curves play a fundamental role in the three-dimensional geometric model (Wu and Wang, 2005; Hao and Bishop, 1997). Curve discretization is a key step to successfully determine the size and distribution of new meshes toward the interior of model. For instance, the currently widely used methods include advancing front method (AFM) (Owen and Saigal, 2015), Delaunay triangulation method (Hao and Bishop, 1997), etc. They all start with discretizing

This work was supported by the National Science Foundation of China under grant number 11772125 and 11472102.

0264-4401
DOI 10.1108/EC-06-2019-0247

the boundary curves, and then the surfaces are meshed conforming to these discretizations (Cuilliere, 1997; Laug and Borouchaki, 2004). Generally, speaking, accurate numerical results depend on high-quality meshes, the high-quality meshes (respecting both density and mesh distortion) can be obtained from an optimal curve discretization.

Previous works in this area have been done by Lohner and Parikh in 1988. The Lohner–Parikh algorithm does not give the same discrete results if we start from one end of the curve or from the other end (Cuilliere, 1997). To avoid the problem that we faced using the Lohner–Parikh algorithm in a three-dimensional adaptive curve discretization procedure, Cuilliere (1997) presented the direct algorithm. He applied a Gaussian integration scheme to compute the total number of discrete segments A_e for three-dimensional parametric curves, and then obtained the parametric coordinates of each discrete node by giving a little parametric increment (Section 3). The direct method is convergent and accurate, but it is time-consuming. Wu combined it with Newton iterative theory to obtain the parametric coordinates of discrete nodes, and thus, its efficiency can be improved (Wu and Wang, 2005). Although the modified direct algorithm reduces the iteration number through Newton iterative theory, it cannot avoid the large computation caused by Gauss integration scheme. Thus, a fast and efficient algorithm is needed for the discretization of three-dimensional parametric curves.

This paper proposes the iterative algorithm. The proposed algorithm computes the parametric increments of all segments to obtain the parametric coordinates of all discrete nodes. This process is recursively applied until the optimal discretization of curves is obtained. The parametric increment of a segment is inversely proportional to the number of sub-segments, which it can be subdivided, and the sum of parametric increments of all segments is constant. Thus, a new expression for the parametric increment of a segment can be obtained. Our algorithm calculates approximately the number of sub-segments, which a segment can be subdivided, thus avoiding Gaussian integration. The computation workload is remarkably reduced.

This paper is organized as follows. Section 2 reviews the basic concepts of geometry and introduces the definition of nodal spacing function. The details of the direct algorithm and its disadvantages are discussed in Section 3. Section 4 gives the details of the new algorithm for adaptive discretization of three-dimensional parametric curves. In Section 5, results are given to compare the proposed algorithm with the direct algorithm. The paper ends with conclusions in Section 6.

2. Basic concepts of geometry and nodal spacing function

In the section, the parametric representation of curves is introduced. It is a common representation type of curves and adopted to ease the process of curve discretization in this paper. In addition, a nodal spacing function is used to represent the expected density distribution of discrete points on a curve. The algorithms of curve discretizations consist of performing an optimal discretization of curves with the density constraint represented by a nodal spacing function.

2.1 Geometrical definition

The B-rep structure provides a description of the CAD model in terms of a set of oriented faces (Chen *et al.*, 2015). Mesh generation for CAD models should require an analytical definition of the surfaces on which the faces are defined and their intersection curves. The analytical definition, which is mathematical representation, should permit us to perform operations, for example, locating a node in space and calculating lengths and tangent vectors of curves, as well as normal vectors and areas of surfaces.

Tensor products of spline functions are the most common form of CAD surface representation (Cuilliere, 1997; Wu and Wang, 2005). Such surfaces can be described by a parametric representation such as:

$$P(u,v) = (x(u,v), y(u,v), z(u,v))$$

$$u_0 \leq u \leq u_1$$

with (1):

$$v_0 \leq v \leq v_1$$

where u and v are the coordinates defined in the parameter domain.

Although the curve is represented in the B-rep as the intersection of two surfaces, the use of such approach for grid generation is not recommended, as it results in an implicit representation of the intersection curve (Thompson, 1999). A more straightforward approach that eases the process of curve discretization is to adopt a parametric representation of the curve that accurately approximates the true intersection. From equation (1), we can get the representation of a parametric curve on the surface, as follows:

$$C(\mathrm{t}) = P\big(u(t), v(t)\big) = \big(x(t), y(t), z(t)\big) \tag{2}$$

When t changes, we get a single parameter curve $C(t)$ on the surface, and $u(t) = u_0 + (u_1 - u_0)t$, $v(t) = v_0 + (v_1 - v_0)t$ and $t \in (0, 1)$.

In the following paragraphs, s corresponds to the arc length along a curve. The arc length of a micro segment on the curve is calculated as follows:

$$ds = \left\| \frac{dC(t)}{dt} \right\| = \sqrt{(dx)^2 + (dy)^2 + (dz)^2} = \sqrt{E\left(\frac{du}{dt}\right)^2 + G\left(\frac{dv}{dt}\right)^2 + 2F\left(\frac{du}{dt}\right)\left(\frac{dv}{dt}\right)}\, dt \tag{3}$$

with E, F and G defined by:

$$E = \frac{\partial P}{\partial u}\frac{\partial P}{\partial u} = \left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 + \left(\frac{dz}{du}\right)^2 \tag{4}$$

$$F = \frac{\partial P}{\partial u}\frac{\partial P}{\partial v} = \frac{dx}{du}\frac{dx}{dv} + \frac{dy}{du}\frac{dy}{dv} + \frac{dz}{du}\frac{dz}{dv} \tag{5}$$

$$G = \frac{\partial P}{\partial v}\frac{\partial P}{\partial v} = \left(\frac{dx}{dv}\right)^2 + \left(\frac{dy}{dv}\right)^2 + \left(\frac{dz}{dv}\right)^2 \tag{6}$$

where E, F and G are the first fundamental (Brockett, 1983) or metric, coefficients of the surface in equation (1).

2.2 Definition of the nodal spacing function

For curve discretization, the nodal spacing function represents the desired arc-length between two adjacent discrete nodes. The key to curve discretization is to perform an optimal discretization of $C(t)$ with the density constraint represented by nodal spacing function.

The nodal spacing function actually used is rarely expressed by explicit formulas, but it is defined on background grids (Owen and Saigal, 2015; Quadros *et al.*, 2010; Deister *et al.*, 2004; Chen *et al.*, 2019). The background grids represent the spatial variation of the distribution of discrete nodes on boundary curves. Each vertex of the background grid stores a size value, which can reflect geometric features of the model, and thus, the background grids are so-called by geometry-based size field. If the background grid is located in the area where the surface curvature and edge curvature of the model is relatively large, then the size values stored on vertexes of the background grid are relatively small. In the process of curve discretization, the size value is used to control the distribution of discrete nodes on curve. Obtaining size value of a node, in the Euclidean space, is a two-step process: Step 1, finding the background gird, which covers the node, and Step 2, obtaining the size value located on this node through interpolating size values stored on all vertexes of the background grid.

The quad-tree and triangular grid are commonly used as background grid in two-dimensional (Owen and Saigal, 2015), and the octree and tetrahedral grid are popularly applied in three-dimensional (Borouchaki *et al.*, 2015; Zhang *et al.*, 2017). In this paper, we use octree as three-dimensional background grid. It is a tree structure that describes three-dimensional space, and thus, it can search the location of one given node quickly, and its time complexity is O(N*log*N) where N is the number of octants (Thompson, 1999; Shephard and Georges, 1991). The octree structure is shown in Figure 1.

The octree has a very wide range of applications in computational geometry. The process of building an octree for a given domain is divided into the following steps: First, we define a cube, named by the root node of tree, which covers the given domain. Then, the cube is subdivided into eight similar shaped octants. Each octant is then examined to determine if it is to be subdivided into eight octants based on given subdivision criteria. If they are to be subdivided, the process of creating the eight octants for that cube is repeated. This subdivision process is recursively applied until the smallest individual octants dose not satisfy the given subdivision criteria. The given subdivision criteria usually depend on the local geometric features of this domain, such as short edges, curvilinear curvature and so on (Thompson, 1999). At last, the octants are subdivided to satisfy the commonly applied one-level difference rule. The one-level difference rule (Thompson, 1999) (also known as the 2:1 rule) is commonly used in octree-based meshing procedures to control mesh gradations and

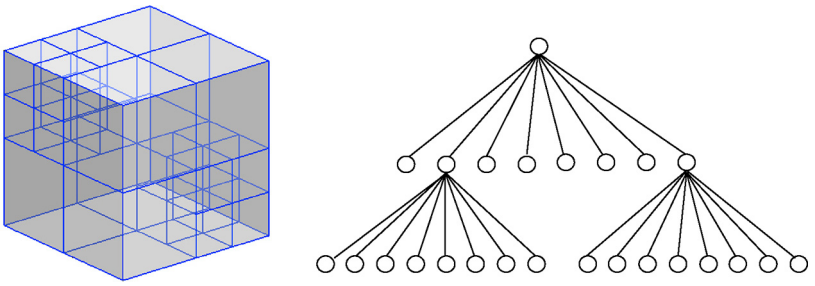


Figure 1.
Octree structure

element aspect ratios. This rule forces octants that share an edge to have no more than a one-level difference. The process of building octree is shown in Figure 2.

3. The direct algorithm

This section simply introduces the direction algorithm previously published by Cuilliere and improved by Wu and Wang. Different from our algorithm, the Gaussian integration scheme is used for obtaining the parametric coordinates of discrete nodes on curve. This requires a lot of computation. The process of the direct algorithm is shown below.

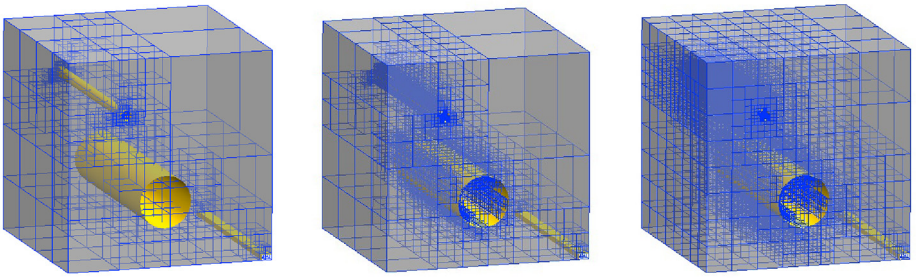
3.1 Calculate the total number of the discrete curve segments

The direct algorithm starts with the computation of the total number of curve segments N_{seg}, which needs to be divided along a parametric curve $C(t)$ with respect to a nodal spacing function $E_d(x, y, z)$. The parameter range of the curve is from t_0 to t_n, and the length of the curve is L.

Consider an interval of length ds at a node $C(t_i)$, and assume that the interval is small enough so that $E_d(x, y, z)$ can be taken to be approximately constant $E_d(x_i, y_i, z_i)$, as shown in Figure 3. Under these assumptions, the number of subdivisions dA_e for the interval will be obtained as:

$$dA_e = \frac{ds}{E_d\left(C(t_i)\right)} = \frac{ds}{E_d(x_i, y_i, z_i)} \tag{7}$$

with



Notes: (a) Octree subdivided by curve curvature; (b) octree subdivided by surface curvature; (c) balance the level of octree

Figure 2.
Process of building octree structure

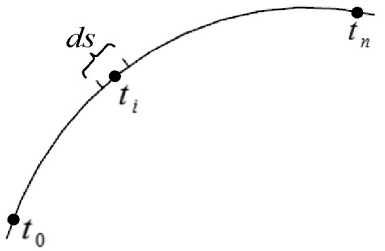


Figure 3.
Micro segment *ds* on the curve

$$C(t_i) = (x_i, y_i, z_i)$$

Under the nodal spacing function $E_d(x, y, z)$, the initial total number of discrete segments A_e is obtained by the curvilinear integral on the whole curve, and can be shown as:

$$A_e = \int_L \frac{ds}{E_d(t)} = \int_{t_0}^{t_n} \frac{1}{E_d(t)} \left\| \frac{dC(t)}{dt} \right\| dt \tag{8}$$

The accuracy of the numerical solution for the integral A_e is of great importance. If the numerical solution for A_e is not sufficiently accurate, the number of nodes to be created is false and the whole process is corrupted. Remarkably, A_e is not an integer in most cases (Cuilliere, 1997; Wu and Wang, 2005), which means that the length of the last segment cannot satisfy the nodal spacing function $E_d(x, y, z)$, as shown in Figure 4, the length of the last segment $t_{n-1}t_n$ is far less than the given nodal spacing function. Then, the real total number of segments N_{seg} is taken to be equal to the nearest integer to A_e, and $E_d(x, y, z)$ is adjusted accordingly so that the error is shared by all discrete segments (Cuilliere, 1997; Wu and Wang, 2005). Thus, the optimal solution can be obtained directly with an increment of a fraction of N_{seg} between a current and the next node equal to:

$$\Delta A_e = \frac{A_e}{N_{seg}} \tag{9}$$

3.2 Calculate the parametric coordinate's t *of discrete nodes on curve*

In fact, the more accurate A_e (or ΔA_e) is, the closer we will be to the optimal solution. A current node generates the next node with the increment ΔA_e, and this process is repeated until all nodes are calculated. The process of implementation is as follows: given a current node $C(t_i)$ at the parametric coordinate t_i, we must find t_{i+1} satisfying the equation (10):

$$\int_{t_i}^{t_{i+1}} \frac{1}{E_d(t)} \left\| \frac{dC(t)}{dt} \right\| dt = \Delta A_e \tag{10}$$

Cuilliere (1997) obtained the parametric coordinate t_{i+1}, through the Gauss integration scheme by giving a little parametric increment δt. This scheme not only increases the computing efforts but also accurate solution cannot be obtained easily (Wu and Wang, 2005). To avoid these shortcomings (Wu and Wang, 2005), applied the Newton iterative theory to the direct method.

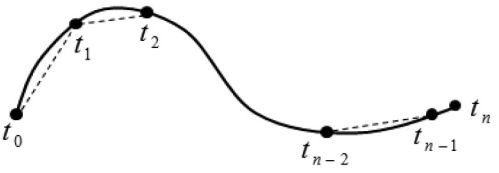


Figure 4.
Curve discretization

Let $f(t) = \int_{t_i}^{t} \frac{1}{E_d(t)} \left\| \frac{dC(t)}{dt} \right\| \mathrm{dt} - \frac{A_e}{N_{seg}} = 0$, then $f'(t) = \frac{1}{E_d(t)} \left\| \frac{dC(t)}{dt} \right\|$. The process of Newton iterative is listed as follows:

- Giving an initial parametric coordinate t_0, $f(t_0)$ and $f(t_0)$ are calculated.
- The parametric coordinate of the current node is used as the initial parametric coordinate of next node. $t_{i+1}^{(0)} = t_i, f\left(t_{i+1}^{(0)}\right) = f(t_i), f'\left(t_{i+1}^{(0)}\right) = f'(t_i)$.
- Iterative sequence is $t_{i+1}^{(k+1)} = t_{i+1}^{(k)} - \frac{f\left(t_{i+1}^{(k)}\right)}{f'\left(t_{i+1}^{(k)}\right)}$, the size of the iterative step is $\lambda = \frac{f\left(t_{i+1}^{(k)}\right)}{f'\left(t_{i+1}^{(k)}\right)}$.
- The iteration is finished if $\left| t_{i+1}^{(k+1)} - t_{i+1}^{(k)} \right| < \varepsilon$ and $t_{i+1} = t_{i+1}^{(k)}$; else $k = k + 1$, return to 2.

where t_{i+1} denotes the ultimate parametric coordinate of the $i+1$-th discrete node on the curve and $t_{i+1}^{(k)}$ denotes the parametric coordinate of the $i+1$-th discrete node at the iteration step k.

The curve will be discretized accurately according to the given nodal spacing function $E_d(x, y, z)$. However, the Gaussian integration scheme (with five points) was used for calculating the parametric coordinate of each discrete node at every iteration step. The great deals of calculation make this algorithm very time-consuming. Thus, a new algorithm named by iterative algorithm is proposed in this paper. It overcomes this disadvantage.

4. Iterative algorithm

The first step of the iterative algorithm is the same as that of the direct algorithm. In the first step, the total number of discrete segments N_{seg} should be calculated for the whole curve. Different from the direct algorithm, the proposed algorithm obtains the parametric coordinates t of all discrete nodes by calculating parametric increments Δt between all adjacent two nodes. During this process, our algorithm replaces Gauss integral scheme with an approximate calculation of integrals. The specific implementation process is shown in the following passage.

To avoid numerical integration, according to the approximate computation of definite integral, equation (10) can be approximated by:

$$\frac{1}{E_d(\overline{t}_i)} \left\| \frac{dC(t)}{dt}\bigg|_{t=\overline{t}_i} \right\| (t_{i+1} - t_i) \approx \Delta A_e \tag{11}$$

with

$$\overline{t}_i = \frac{(t_i + t_{i+1})}{2} \tag{12}$$

Equation (11) can be rewritten as:

$$\Delta t_i \approx (\Delta A_e)/J_i \tag{13}$$

with Δt_i and J_i defined by:

$$\Delta t_i = t_{i+1} - t_i \tag{14}$$

$$J_i = \frac{1}{E_{\mathrm{d}}(\overline{t}_i)} \left\| \frac{dC(t)}{dt}\Big|_{t=\overline{t}_i} \right\| \tag{15}$$

J_i refers to the number of sub-segments, which the micro segment at the node $C(\overline{t}_i)$ is subdivided according to $E_{\mathrm{d}}(\overline{t}_i)$. As ΔA_e is a constant, Δt_i is in the direct ratio of $\frac{1}{J_i}$. Because of $\sum_{i=0}^{n-1} \Delta t_i = 1$, Δt_i can be defined as:

$$\Delta t_i = \frac{\frac{1}{J_i}}{\sum_{i=0}^{n-1}\left(\frac{1}{J_i}\right)} \tag{16}$$

To obtain parametric increments between all adjacent two nodes, the iterative process is needed, and it is as follows:

- The total number of discrete segments N_{seg} on the whole edge can be calculated by equation (11). Thus, the parametric increments, at the first iteration step, are defined as:

$$\Delta t_i^{(0)} = \frac{1}{N_{seg}},\ i = 0, 1, \ldots N_{seg} - 1 \tag{17}$$

Equation (17) means that the parametric coordinates of all discrete nodes on the whole edge, at the first iteration step, are:

$$t_i^{(0)} = \frac{i}{N_{seg}}, i = 1, \ldots N_{seg} - 1 \tag{18}$$

All discrete nodes, at the first iteration step, are uniform distribution, as shown in Figure 5.

- According to equation (16), the parametric increments, at the iteration step $k + 1$, are defined as:

$$\Delta t_i^{(k+1)} = \frac{\frac{1}{J_i^{(k)}}}{\sum_{i=0}^{n-1}\left(\frac{1}{J_i^{(k)}}\right)}, i = 1, 2, \ldots N_{seg} - 1 \tag{19}$$

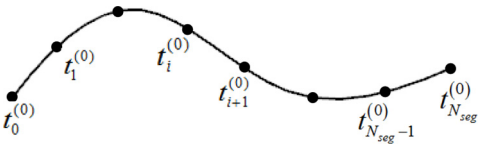


Figure 5.
Equidistant distribution of nodes at the first iteration step

with

$$J_i^{(k+1)} = \frac{1}{E_{\mathrm{d}}\left(\overline{t_i^{(k)}}\right)} \left\| \left. \frac{ds}{dt} \right|_{t=\overline{t_i^{(k)}}} \right\| \tag{20}$$

where k is the number of iteration steps. At the iteration step $k + 1$, the parametric coordinate's t of all discrete nodes on the whole edge is:

$$t_i^{(k+1)} = \sum_{m=0}^{i-1} \Delta t_m^{(k+1)}, i = 1, 2, \ldots N_{seg} - 1 \tag{21}$$

The discrete nodes are non-uniform distribution according to nodal spacing function, as shown in the following diagram (Figure 6).

- The iteration is finished. If $\left| t_i^{(k+1)} - t_i^{(k)} \right| < \varepsilon, i = 1, 2, \ldots N_{seg}-1$, and $t_i = t_i^{(k+1)}$, $i = 1, 2, \ldots N_{seg}-1$; else $k = k + 1$, return to 2.

It can be seen from our algorithm that the parametric coordinates of all discrete nodes are obtained at each iteration step, but they may not be the final parametric coordinates. The iterative process continues until the parametric coordinates of all discrete nodes have little change, and then the final parametric coordinates of all discrete nodes are obtained. By replacing the Gauss integral process with the approximate calculation of integral, our algorithm avoids numerical integral in the process of calculating the parametric coordinates of all discrete nodes, and thus, our algorithm consumes less CPU time. In the following section, several examples are given to illustrate our algorithm is more efficient than the direct algorithm.

5. Results and comparisons

In this section, our algorithm is compared with the direct algorithm. We measure performance mainly in the accuracy of curve discretizations and its required CPU time (in milliseconds). The accuracy of a curve discretization, with respect to a nodal spacing function, is defined from the distribution of a normalized quality factor Q (Cuilliere, 1997). The CPU time, on an Intel Xeon processor 1.6 GHz, only includes the time required by boundary curves discretization, and does not include the time consumption of building background grids. Because the direct algorithm is the same as our algorithm for creating background grids, so we only compare the differences in the CPU time required by curve discretizations. It can be seen from numerical examples that these two algorithms can both obtain an optimal discretization of curves in the context of variations in the nodal spacing function along the curves. However, compared to the direct algorithm, the more complex the model, the more time our algorithm saves. In addition, the qualities of meshes conforming to curves discretization results are given to illustrate that meshes are suitable for subsequent numerical analysis, such as FEM (Nayroles *et al.*, 1992) and BEM (Qin *et al.*, 2010; Brebbia *et al.*, 1984; Zhang *et al.*, 2017).

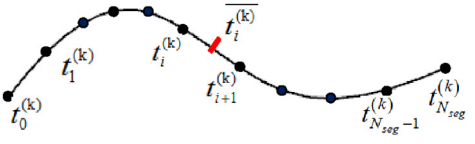


Figure 6.
Non-uniformly distribution of nodes at iteration step k

The accuracy of a curve discretization, with respect to a nodal spacing function, is defined from the distribution of a normalized quality factor Q (Cuilliere, 1997). The normalized quality factor Q_i for the i-th curvilinear segment is:

$$Q_i = \frac{\int_{L_i} \frac{ds}{E_d(t)}}{\Delta A_e} = \frac{\int_{t_i}^{t_{i+1}} \frac{1}{E_d(t)} \left\| \frac{dC(t)}{dt} \right\| dt}{\Delta A_e}$$

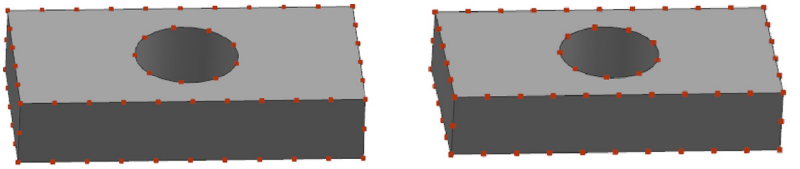


Figure 7. Uniform discretization results for regular curves

Notes: (a) Iterative algorithm; (b) direct algorithm

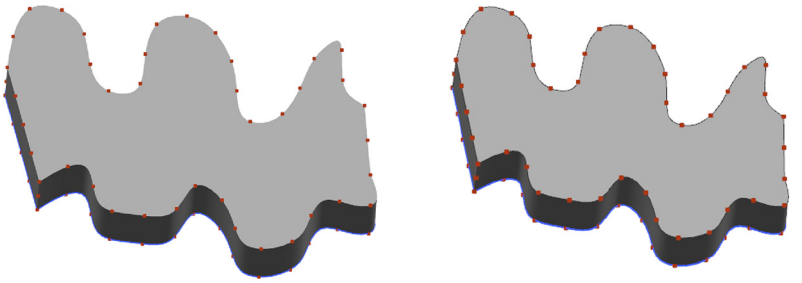


Figure 8. Uniform discretization results for irregular curves

Notes: (a) Iterative algorithm; (b) direct algorithm

Table I. Statistics of regular curves under uniform size field

Method	Quality factor of curve discretization M	σ	Time (ms)
Iterative algorithm	1.0	0.0	2
Direct algorithm	1.0	0.0	11

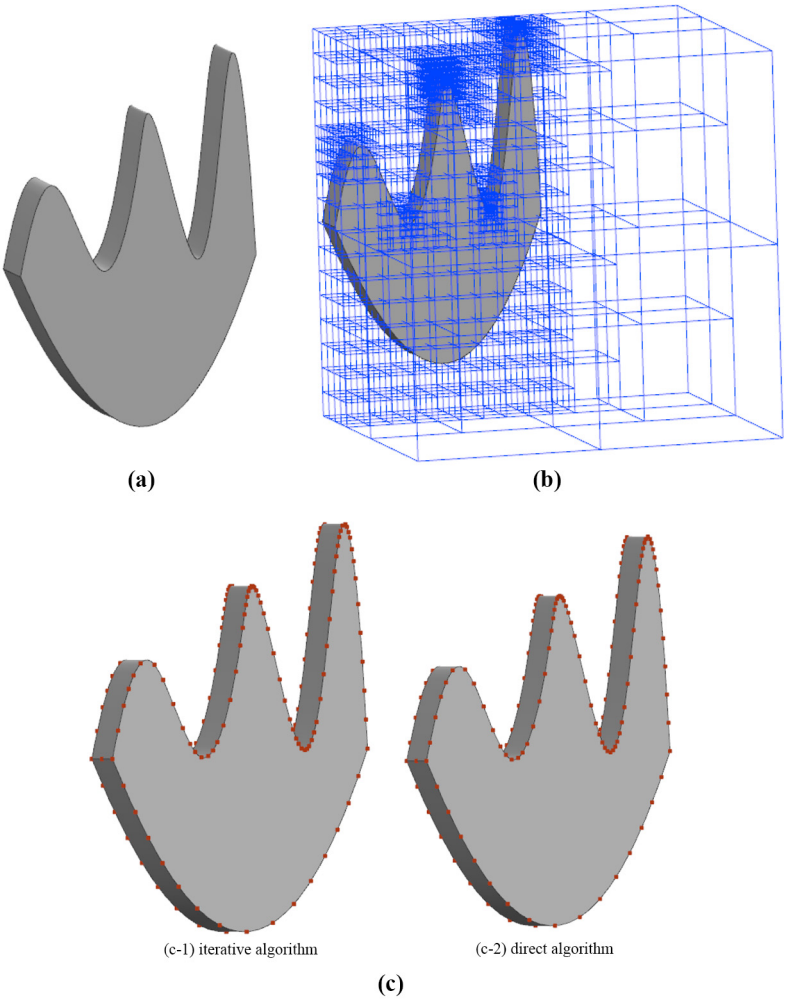
Table II. Statistics of irregular curves under uniform size field

Method	Quality factor of curve discretization M	σ	Time (ms)
Iterative algorithm	0.999131	0.006726	70
Direct algorithm	1.000131	0.000930	358

The optimal solution occurs when all segments have a Q_i equal to 1 [Equation (10)], and the mean value for all segments is [Equation (8)] (Cuilliere, 1997):

$$\overline{Q} = \frac{\sum_{i=1}^{N_{seg}} Q_i}{N_{seg}} = \frac{1}{N_{seg}} \sum_{i=1}^{N_{seg}} \frac{N_{seg}}{A_e} \int_{L_i} \frac{ds}{E_d(s)} = \frac{1}{A_e} \int_{L} \frac{ds}{E_d(s)} = 1.0$$

The mean M and standard deviation σ of the normalized quality factors are computed, and they are used to assess the accuracy of a curve discretization.



Notes: (a) Model with sharp curves; (b) octree size field; (c) discretization results of boundary curves

Figure 9.
Non-uniform discretization results for sharp curves with large curvature

The anther quality evaluation indicators are introduced to assess the mesh quality. For a triangle ABC, α quality is used to evaluate the quality of one individual element, β denotes the average value of element qualities and δ denotes an excellent rate. The α quality is in the range from 0 to 1. The bigger α quality of one element, the more regular its shape is. If α quality of one individual element is greater than 0.8, this element is excellent. The equations (22) and (23) for calculations of α and β is as follows:

$$\alpha = 2\sqrt{3}\frac{\|CA \times CB\|}{\|CA\|^2 + \|AB\|^2 + \|BC\|^2} \tag{22}$$

$$\beta = \frac{\sum_{i=1}^{NT} \alpha_i}{NT} \tag{23}$$

where NT is the number of all elements. In the ideal case, $\alpha = 1$, and $\alpha = 0$ if A, B and C are co-linear.

Numerical examples are divided into two categories according to nodal spacing function: those whose boundary curves are uniformly discrete, and those whose boundary curves are non-uniformly discretized. When a nodal spacing function $E_d(x, y, z)$ is a constant that is to say that the size values stored on the vertexes of background grids are the constant, boundary curves will be uniformly discretized. If a nodal spacing function $E_d(x, y, z)$ gets different size values through background grids such as octree, boundary curves will be non-uniformly discretized.

5.1 Uniform discretization of curves

The examples in this section are designed to illustrate that the proposed algorithm can be applied to get uniform discretizations of curves with respect to a nodal spacing function, which is constant.

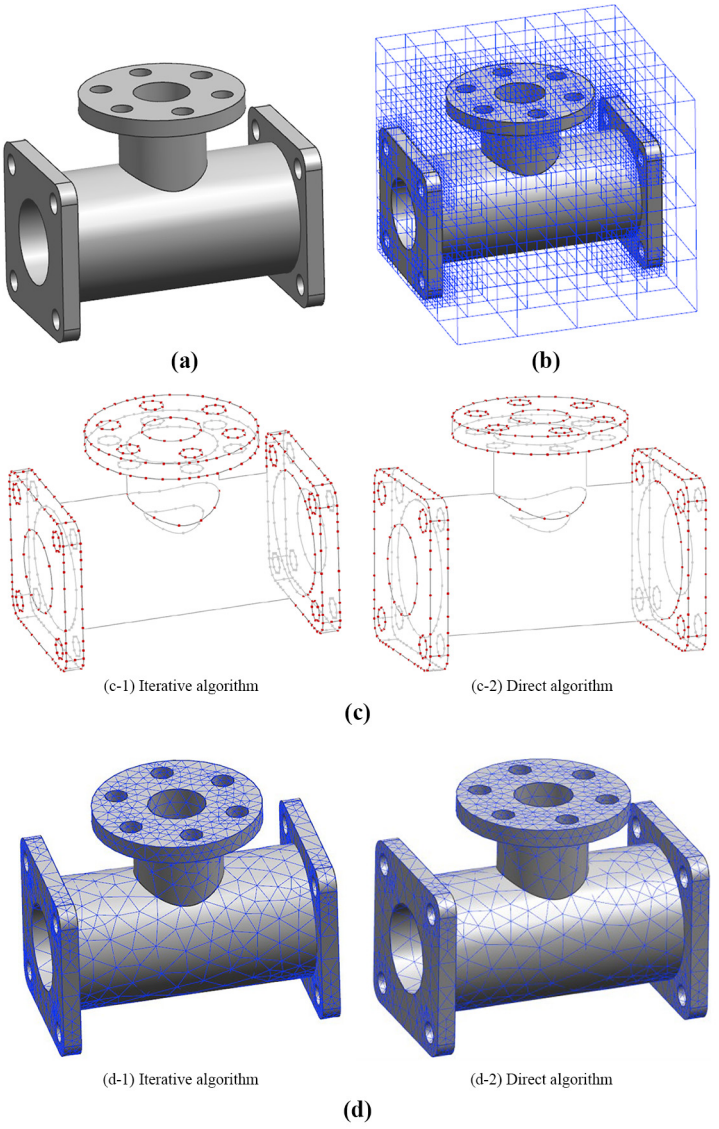
Different types of curves are used to test the efficiency of two discrete algorithms under the uniform size field, in which the nodal spacing function $E_d(x, y, z)$ is constant. Figure 7 shows the model whose boundary curves are regular curves, such as straight-lines and arcs. Figure 8 shows the model whose boundary curves are irregular curves such as B-spline curves. These two figures illustrate that these two algorithms have almost the same boundary curves discretization results. As can be seen from Tables I and II, the mean (M) of normalized quality factors (Q) is approximately equal to 1, their standard deviation at least reaches to the value at the order of the magnitude of 10^{-3}. These data illustrate that the normalized quality factors of every discrete segment is approximately equal to 1. Then, these two algorithms can perform the uniform discretization of curves with respect to the uniform size field. However, through Tables I and II, it is obvious that our algorithm is more time-saving than the direct algorithm.

Table III. Statistics of sharp curves under non-uniform size field

Method	Quality factor of curve discretization		Time (ms)
	M	σ	
Iterative algorithm	0.997578	0.085362	55
Direct algorithm	0.998938	0.016630	431

5.2 Non-uniform discretization of curves

This section is presented to demonstrate that the proposed algorithm can be competent to obtain an optimal discretization of curves in the context of variations in the nodal spacing function along the curves. What is more, the engineering models are given to verify that the proposed algorithm can be applied in practice. Then, the more complex the model, the more time our algorithm saves in the process of curve discretization.



Notes: (a) Three-way pipe; (b) octree size field; (c) discretization results of boundary curves; (d) mesh generation on model surface

Figure 10.
Three-way pipe

This example is used to show the iterative algorithm can be applied to the discretization of curves in the context of steep variations in a nodal spacing function along the curves. The discrete model with sharp curves is shown in Figure 9(a). Figure 9(b) indicates the steep variations in the nodal spacing function along the boundary curves of model. The variations in nodal spacing function are represented by the variations in the density of octree cells. As can be seen from Figure 9(b), the node spacing function varies greatly as it

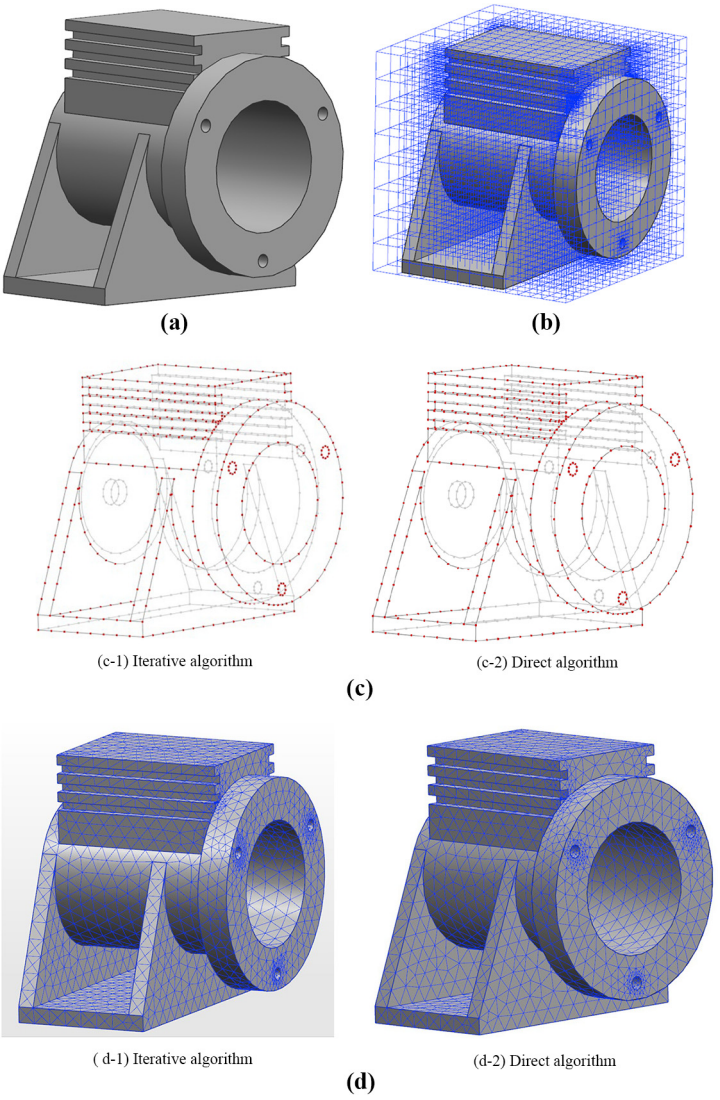


Figure 11.
Engine case

Notes: (a) Engine case; (b) octree size field; (c) discretization results of boundary curves; (d) mesh generation on model surface

approaches the portions of the curve with large curvature. Figure 9(c-1) shows the curve discretization result of the iterative algorithm. More discrete nodes are concentrated in those portions of the curve with large curvature. What is more, Table III shows that the mean (M) of normalized quality factors obtained by the iterative algorithm is closed to 1. The standard deviation (σ) of the normalized quality factors reach the value at the order of the magnitude of 10^{-2}. These data illustrate that the normalized quality factors of every discrete segment is approximately equal to 1. Then, the proposed algorithm can perform the optimal discretization of curves with respect to the prescribed spacing function.

In addition, Figure 9(c) reveals that the proposed algorithm and the direct algorithm have nearly the same discretization results of boundary curves. This can also be demonstrated by the data in Table III. The magnitude of the difference in the quality factor between the two algorithms is small. However, the proposed algorithm requires less CPU time than the direct algorithm.

Figures 10(a) and 11(a) show the three-way pipe and the engine case, respectively. Most of their boundary curves are regular curves, such as straight-line, arc and ellipse lines. As the analytic expression for the equation of the regular curves can be obtained, the derivative of a point on the curve is easy to compute.

The octree size field is showed in Figures 10(b) and 11(b). The variations in the density of octree grids represent the variations in nodal spacing function. Figures 10(c) and 11(c) imply that the proposed algorithm and the direct algorithm have nearly the same discretization results of boundary curves. Moreover, as can be seen from the statistics in Tables IV and V, the means of normalized quality factors of these two algorithms are both approximately equal to 1. Then, the standard deviations of normalized quality factors are both small. Thus, the normalized quality factors of every discrete segment are approximately equal to 1. The results illustrate that these two algorithms can both be applied to perform the optimal discretization of curves with respect to the prescribed spacing function. However, the proposed algorithm requires less CPU time than the direct algorithm. Though the standard deviation of normalized quality factors of the proposed algorithm is slightly larger than that of the direct algorithm, the mesh quality factors illustrate that the minor differences in standard deviation between the two algorithms do not affect the quality of the subsequently generated mesh. Figures 10(d) and 11(d) show the mesh generated by AFM based on the

Table IV. Three-way pipe statistics

Method	Quality factor of curve discretization M	σ	Mesh quality factor β	δ	Time (ms)
Iterative algorithm	0.997985	0.067912	0.9326	0.9862	27
Direct algorithm	0.998657	0.016157	0.9326	0.9880	327

Table V. Engine case statistics

Method	Quality factor of curve discretization M	σ	Mesh quality factor β	δ	Time (ms)
Iterative algorithm	0.998495	0.051912	0.8782	0.9712	58
Direct algorithm	1.000916	0.012583	0.8761	0.9703	369

above discretization results of boundary curves. The statistics in Tables IV and V illustrate that the mesh qualities are almost similar, and meshes can be suitable for subsequent numerical calculations.

Figures 12(a) and 13(a) show a camshaft and a centrifugal compressor impeller, respectively. Most of boundary curves of these models are B-spline curves and are more complex than that of the models shown in Figures 10 and 11. In Figures 12(c) and 13(c), two different discrete algorithms can adaptively discretize the boundary curves of these models based on the prescribed spacing function represented by octree size fields. It can be seen that the discrete nodes, in the portions of curves with lager curvature, are intensive, while the discrete nodes located in the gentle portions of curves are relatively sparse.

As can be deduced from quality factors of curve discretization in Tables VI and VII, the normalized quality factors of every segment are both approximately equal to 1. This illustrates that these two algorithms can perform an optimal discretization of the curve with respect to the prescribed spacing function. Though the discretization results obtained by these two algorithms are almost the same, the time required by the proposed algorithm is significantly less than the direct algorithm.

Figures 12(d) and 13(d) show the meshes generated by AFM based on the above discretization of boundary curves. As the discretization of boundary curves are used as input of mesh generation, the final generated meshes are also adaptive distribution. The

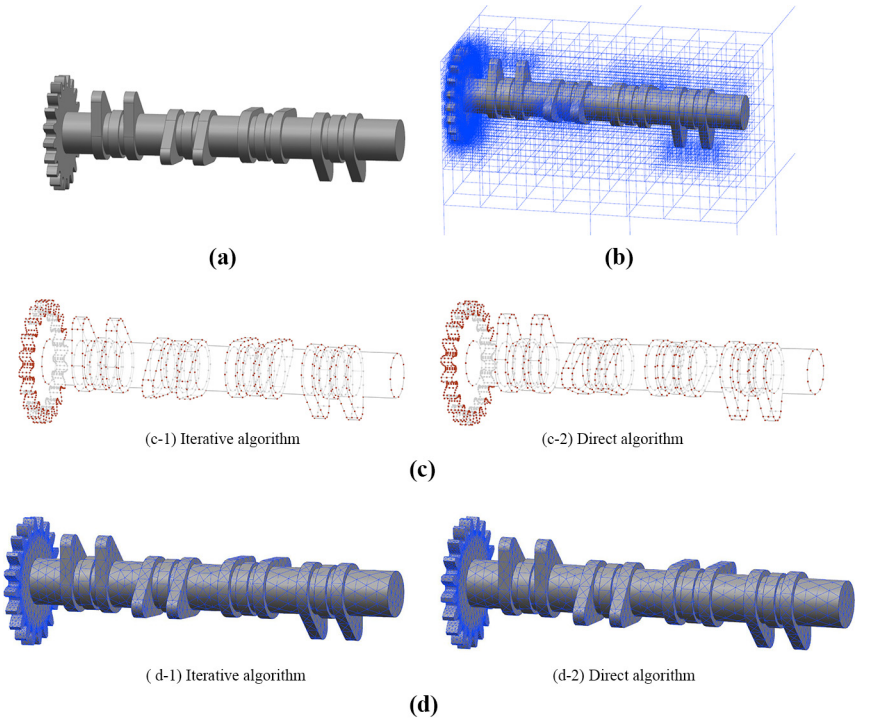


Figure 12.
The camshaft

Notes: (a) Engine case; (b) octree size field; (c) discretization results of boundary curves (d) mesh generation on model surface

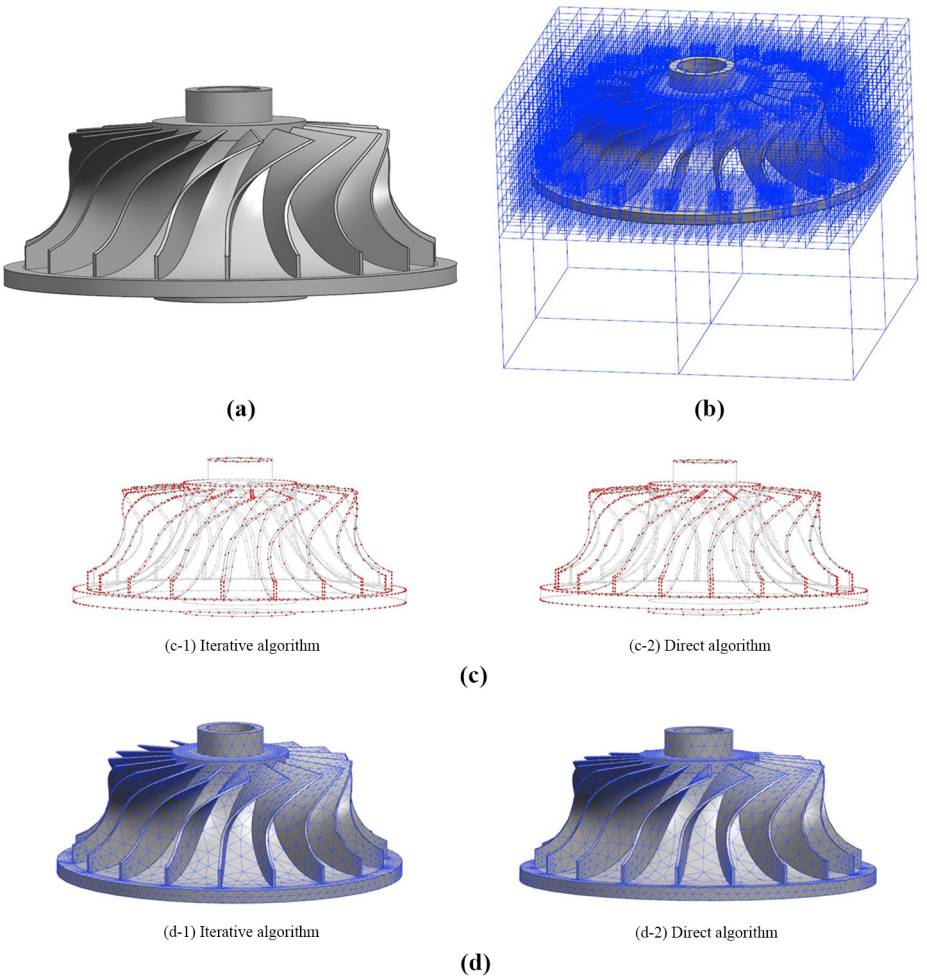


Figure 13. Centrifugal compressor impelle

Notes: (a) Centrifugal compressor impeller; (b) octree size field; (c) discretization results of boundary curves; (d) mesh generation on model surface

Method	Quality factor of curve discretization M	σ	Mesh quality factor β	δ	Time (ms)
Iterative algorithm	0.996587	0.047374	0.8605	0.9011	309
Direct algorithm	0.999547	0.012597	0.8640	0.9026	2044

Table VI. Statistics of camshaft

average mesh quality factor β and the excellent rate δ, showed in Tables VI and VII, demonstrates that the qualities of most meshes are acceptable, and meshes are conductive to subsequent numerical analysis. Thus, the proposed algorithm can be applied to mesh generation, and save time spent on preprocessing.

From Tables IV-VII, we can see that the time of the proposed method is less than 20 per cent of the time of the direct method. Compared with the direct method, the more complex the model, the more time the proposed method saves. A complex model means that the number of splines in boundary curves is large and the length of splines is long. According to the above models, the time saving of the proposed algorithm, relative to the direct algorithm, and the type statistics of the boundary curves are listed in Table VIII.

As can be seen from Table VIII, the time saved for three-way pipe and engine case does not seem to be a lot. This is because the boundary curves of these two models are almost all regular curves, such as straight lines and arcs, and the total time spent on the curve discretization is less. However, as can be verified from Figure 12 and Table VIII, the boundary curves of the camshaft include many short splines, and our algorithm saves 1.735 s as compared with the direct method. Compared to the camshaft, the centrifugal compressor impeller owns many long splines, and the time saved by proposed algorithm increases to 8.463 s. The results demonstrated that the more complex the model, the more time the proposed algorithm saves in the process of curves discretization. This is because that in the direct algorithm, the length of curves is obtained by Gaussian integration, and this process is time-consuming, especially for spline curves. However, the iterative algorithm uses the approximate computation of definite integral instead of Gaussian integration.

6. Conclusion

In this paper, we have proposed the iterative algorithm. The proposed algorithm computes the parametric increments of all segments to obtain the parametric coordinates of all discrete nodes. This process is recursively applied until the optimal discretization of curves is obtained. A new expression for the parametric increment of a segment is derived in this paper. Then, the number of sub-segments, which a segment can be subdivided is calculated approximately, thus avoiding Gaussian integration. A number of numerical examples with different geometric curves confirm that the proposed algorithm can be applied to perform an

Table VII. Statistics of centrifugal compressor impeller

Method	Quality factor of curve discretization		Mesh quality factor		Time (ms)
	M	σ	β	δ	
Iterative algorithm	1.001535	0.081816	0.8323	0.8039	1338
Direct algorithm	0.999547	0.012597	0.8351	0.8036	9801

Table VIII. Type statistics of the boundary curves and time saving comparison

Model	Time saving (s)	No. of splines	No. of regular lines
Three-way pipe	0.3	2	84
Engine case	0.311	0	151
Camshaft	1.735	200	290
Centrifugal compressor impeller	8.463	198	99

optimal discretization of curves in the context of variations in a nodal spacing function along the curves. The proposed algorithm has the same discretizations of boundary curves as the direct algorithm while consuming less CPU time. It is worth noting that the more complex the model, the more time is saved by the proposed algorithm in the process of curve discretization. The proposed algorithm has been successfully applied in mesh generation, and the results of curves discretization can meet requirements for mesh generation. This further illustrates the effectiveness of the proposed algorithm and improves the efficiency of preprocessing of numerical simulation.

References

Borouchaki, H., Laug, P. and George, P. (2015), "Parametric surface meshing using a combined advancing-front generalized Delaunay approach", *International Journal for Numerical Methods in Engineering*, Vol. 49 Nos 1/2, pp. 233-259.

Bournival, S., Cuillière, J.C. and François, V. (2010), "A mesh-geometry based method for coupling 1D and 3D elements", *Advances in Engineering Software*, Vol. 41 No. 6, pp. 838-858.

Brebbia, C.A., Tells, J.C.F. and Wrobel, L.C. (1984), "Boundary element techniques".

Brockett, R.W. (1983), *Differential Geometric Control Theory*, Birkhauser.

Chen, J., Cao, B., Zheng, Y., Xie, L., Li, C. and Xiao, Z. (2015), "Automatic surface repairing, defeaturing and meshing algorithms based on an extended B-rep", *Advances in Engineering Software*, Vol. 86, pp. 55-69.

Chen, J., Xiao, Z., Zheng, Y., Zheng, J., Li, C. and Liang, K., (2019), "Automatic sizing functions for unstructured surface mesh generation", *International Journal for Numerical Methods in Engineering*, p. 109.

Cuilliere, J.C. (1997), "A direct method for the automatic discretization of 3D parametric curves", *Computer-Aided Design*, Vol. 29 No. 9, pp. 639-647.

Cuillière, J.C. (1998), "An adaptive method for the automatic triangulation of 3D parametric surfaces", *Computer-Aided Design*, Vol. 30 No. 2, pp. 139-149.

Cunha, A., Canann, S. and Saigal, S. (1997), "Automatic boundary sizing for 2d and 3d meshes", *In Trends in Unstructured Mesh Generation*, AMD-220, Vol. 220, pp. 65-72.

Deister, F., Tremel, U., Hassan, O. and Weatherill, N.P., (2004), "Fully automatic and fast mesh size specification for unstructured mesh generation", *Engineering with Computers*, Vol. 20 No. 3, pp. 237-248.

Hao, C. and Bishop, J. (1997), "Delaunay triangulation for curved surfaces", *International Meshing Roundtable Proceedings*, pp. 115-127.

Laug, P. and Borouchaki, H. (2004), "Curve linearization and discretization for meshing composite parametric surfaces", *Communications in Numerical Methods in Engineering*, Vol. 20 No. 11, pp. 869-876.

Nayroles, B., Touzot, G. and Villon, P. (1992), "Generalizing the finite element method: diffuse approximation and diffuse elements", *Computational Mechanics*, Vol. 10 No. 5, pp. 307-318.

Owen, S.J. and Saigal, S. (2015), "Surface mesh sizing control", *International Journal for Numerical Methods in Engineering*, Vol. 47 Nos 1/3, pp. 497-511.

Qin, X., Zhang, J., Li, G., Sheng, X., Song, Q. and Mu, D., (2010), "An element implementation of the boundary face method for 3D potential problems", *Engineering Analysis with Boundary Elements*, Vol. 34 No. 11, pp. 934-943.

Quadros, W.R., Vyas, V., Brewer, M., Owen, S.J. and Shimada, K. (2010), "A computational framework for automating generation of sizing function in assembly meshing via disconnected skeletons", *Engineering with Computers*, Vol. 26 No. 3, pp. 231-247.

Shephard, M.S. and Georges, M.K. (1991), "Automatic three-dimensional mesh generation by the finite octree technique", *International Journal for Numerical Methods in Engineering*, Vol. 32 No. 4, pp. 709-749.

Thompson, J.F. (1999), *Handbook of Grid Generation*, CRC Press, pp. 47-52.

Wu, B. and Wang, S. (2005), "Automatic triangulation over three-dimensional parametric surfaces based on advancing front method", *Finite Elements in Analysis and Design*, Vol. 41 Nos 9/10, pp. 892-910.

Zhang, J., Lin, W., Dong, Y. and Ju, C., (2017), "A double-layer interpolation method for implementation of BEM analysis of problems in potential theory", *Applied Mathematical Modelling*, p. 51.

Corresponding author
Jianming Zhang can be contacted at: zhangjianm@gmail.com