Computer-Aided Design 79 (2016) 36–47

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Reconsideration of T-spline data models and their exchanges using STEP

Wenlei Xiao [a,b], Yazui Liu [a], Rui Li [a], Wei Wang [a], Jianmin Zheng [c], Gang Zhao [a,b,*]

[a] School of Mechanical Engineering & Automation, 100191 Beihang University, Beijing, China
[b] MIIT Key Laboratory of Aeronautics Intelligent Manufacturing, 100191 Beihang University, Beijing, China
[c] School of Computer Engineering, 639798 Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 27 January 2016
Accepted 4 June 2016

Keywords:
T-spline
Data model
STEP
Data exchange

ABSTRACT

T-spline is a new approach to define freeform surfaces with relatively less control points than NURBS and is able to represent a model using a single surface without joining errors. Whereas, the complexity of T-spline data models leads numerous difficulties in its programming, which hinders the research and development of T-spline technologies. In addition, the data exchange of T-spline models still remains on a primitive level, and no standardized data format has been published so far. This article gives a reconsideration to the existing T-spline definitions, and proposes a set of redesigned data models which have much more understanding conveniences to both human and computer. Moreover, STEP-compliant data models are designed using the proposed T-spline models to standardize their data exchange between different CAx systems. The combination of T-spline with other product models in ISO 10303 makes it convenient to exchange the versatile resource data in a hybrid neutral file. A prototype system is developed for the validation purpose, which consists of a TSM-to-STEP convertor, a STEP parser and a T-spline kernel. Using the developed prototype system, one can automatically convert a Rhino system exported TSM file to a STEP file in the P21 format, which can be then parsed using the STEP reader and processed by the T-spline kernel. Some testing examples show that the proposed data models are much more efficient in processing and exchanging the T-spline data.

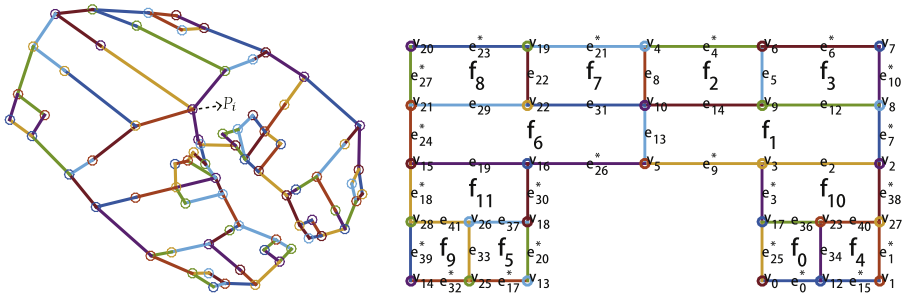
1. Introduction

T-spline has attracted great interests from researchers since its emergency in 2003 [1]. Comparing to NURBS, it has great advantages of less control points, localized refinement and tessellation operations [2,3] and isogeometric analysis [4,5]. In addition, T-spline has shown its progressively powerful modeling functions comparing to NURBS, especially after *Rhino*© releases the T-spline plug-in [6]. The success of T-spline kernel in Rhino has shown an optimistic prospect on integrating T-spline into other CAx systems. For example, a CAD system can provide to the users another modeling method via T-spline, a CAE system can introduce a new basis for isogeometric analysis [7,8], a CAM system may support a novel path planning ability to generate a five-axis machining path for a whole part directly [9], and a CNC system could use a T-spline model as its precisely defined workpiece part for object-oriented and inspection based closed-loop manufacturing [10], which as well obeys the concept of STEP-CNC [11–13]. Therefore, it can be considered that more and more CAx systems will provide this new modeling method in the future. Regarding the prosperous development of T-spline, it will grow up to be a necessity to exchange T-spline models between different CAx systems, just alike the requisite of other conventional B-Rep models. In order to fulfill this request, *Rhino*© has recently unfolded a text-based TSM (T-spline Mesh) file format [14] for storing its exported T-spline data. However, the practical use has proved that the analyticity of TSM is a long way from satisfactory for complex data exchange. Developers generally just have to spend a lot of time and efforts in developing a data parser, before they really can import a T-spline model generated by *Rhino*©. In order to solve this dilemma, the standardization of T-spline models has to be implemented before miscellaneous customized definitions flood the research and development fields.

Standardized T-spline models must be compact for storing, flexible for data defining, and reversible for indexing, as data

* Corresponding author at: School of Mechanical Engineering & Automation, 100191 Beihang University, Beijing, China.
E-mail address: zhaog@buaa.edu.cn (G. Zhao).

http://dx.doi.org/10.1016/j.cad.2016.06.004
0010-4485/



(a) A T-mesh example. (b) Pre-image of the T-mesh.

Fig. 1. A T-mesh example and its pre-image.

structures of T-spline are much more complex than that of NURBS. C. Asche et al. proposed a set of efficient data structures for T-spline modeling [15], which rely on the half-edge data structures in the CGAL geometry library [16], whereas the obtained results are still on a primary level, for they still use the conventional two-layer T-spline data models (the T-mesh and its pre-image). Lin et al. introduced the extended T-mesh data structures that make it more easier for computation, but brings many redundant data [17]. Very few more research literatures can be found on the similar topic. The majority of T-spline researchers are obliged to develop their own T-spline kernel from scratch without enough consideration on the efficiency and operability. This status has hindered the development of T-spline for a long time, and drove many new researchers away.

Under those considerations, this paper conducts great efforts on T-spline data modeling using an object-oriented data modeling framework. The conventional T-mesh data structure is decomposed into the parametric, topological and Cartesian layers (so called three-layer models), which have great advantages in storing, accessing and operating the T-spline data. Object-oriented T-spline data models are redesigned to obtain more conveniences to both human and computer, and the STEP-compliant data exchange format is subsequently derived using the EXPRESS modeling language [18], and programmed using the SDAI method [19]. The conventional ISO 10303 standard is thereby extended. Using the STEP standard, NURBS has been successfully modeled, and widely used in the forms of AP203 or AP214 files for CAD data exchanges between different CAx systems. Although STEP has regarded NURBS as the major modeling method for freeform surfaces [20,21], the strive on modeling T-spline using STEP extends the compatibility of the STEP standard to manifold CAD models, let alone the numerous advantages of T-spline against NURBS [1]. If the T-spline models can be supported together with B-Rep models, the data exchange ability can be significantly enhanced and the exchange errors can be managed in a reasonable manner. Since the geometric data models are the base for other applications and implementations, the addition of T-spline extends and changes the way of many other STEP-compliant CAx applications as well.

2. Reconsideration of T-spline data models

This section studies the data structures of T-spline. Firstly, a brief review of T-spline and T-mesh is taken to discuss the complexities and difficulties in handling the existing T-spline data models. Secondly, a set of redesigned T-spline data models are proposed, which organize the T-spline data structures into three layers. The data structures are described and discussed using a simple T-spline example.

2.1. Review of T-spline and T-mesh

Recently, most researches have defined a T-spline surface by means of a control grid called T-mesh [2,22], which provides information in both Cartesian and parametric spaces (control points in the Cartesian space and the pre-image in the (s, t) parametric space). This definition is basically extended from that of the tensor-product B-spline surfaces, which uses a rectangular grid of control points. In order to describe the parametric space, the pre-image of a T-mesh is presented as an attachment to the T-mesh. Fig. 1 shows a so designed T-mesh and its pre-image. These definitions reluctantly work for the theoretical description, whereas bring a lot of troubles in theory understanding, data exchanging and software programming. The difficulties imply that the introduced T-mesh from the control grid of B-spline is not sufficient for many extraordinary properties of T-spline. There are essentially three major drawbacks in the existing T-mesh definitions:

(1) T-mesh contains not only Cartesian but also parametric data, which bring great complexities in operating the data structures.

In order to calculate the equation of a point based T-spline (instead of grid based) [1]

$$P(s,t) = \frac{\sum_{i=1}^{n} P_i B_i(s,t) w_i}{\sum_{i=1}^{n} B_i(s,t) w_i} \tag{1}$$

there are two vital variables that need to be determined in advance: P_i and $B_i(s, t)$. Wherein, P_i are the control points, and $B_i(s, t)$ are the basis functions given by

$$B_i(s,t) = N_{i0}^m(s) N_{i0}^n(t) \tag{2}$$

where N_{i0}^m and N_{i0}^n are the mth and nth B-spline basis functions associated with individual knot vectors in s and t directions. Usually $m = n = 3$, so that they become cubic B-splines. Thus, to specify a T-spline, one must provide a pair of knot vectors for each control point.

An intuitive approach is to store a pair of knot vectors in each control point of the T-mesh. However, this approach consumes additional storing space for redundant data, as most knot vector components are the same in adjacent control points. Moreover, the locality of T-spline causes difficulties in determining which control points should be involved in calculating a specified parametric coordinate (s, t). This is because the T-spline equation (Eq. (1)) is point based rather than grid based, so that the involved control points cannot be easily determined from the T-mesh grid, especially when multiple knots occur in the T-mesh.

(2) The complexity of the pre-image of a T-mesh is ignored significantly, although it contains the most abundant information.

Table 1
Terminologies defined in the T-mesh.

Terms	Descriptions
Vertex	A coordinate (s, t) in the parametric space.
Edge	Determined by start and end vertices.
Face	Surrounded by a loop of oriented edges.
Point	The location of a vertex in the Cartesian space.
Wedge	A square of vertices.

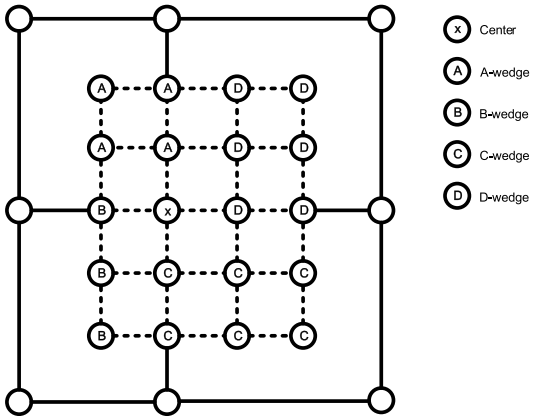


Fig. 2. Definition of the vertex multiplicity as four wedges in the TSM file format [14].

Table 1 provides a summary of terminologies defined in the T-mesh.

The pre-image of a T-mesh, without even a proprietary name, contains the most abundant data elements. As shown in Fig. 1(b), the pre-image consists of vertices, edges and faces, which all can have possibilities of multiplicity. The multiplicity of T-spline significantly differs from that of B-spline. Since the T-mesh is not a regular control grid, the multiplicity of each vertex need to be defined individually by wedges constituted by several duplicated vertices. Fig. 2 shows an exemplary definition of the wedges in the TSM file format [14]. The number of wedges is the same as the functional valence of the vertex. The complexity brought by the multiplicity causes great difficulties in designing data models and implementing algorithms, in which the pair of corresponding knot vectors usually have to be deduced from the pre-image of a T-mesh.

Another notation that is hard to understand and difficult to program is the ambiguity of a vertex. On one hand, a vertex represents one or more Cartesian points, as it belongs to the pre-image of the T-mesh, and several distinct control points can be mapped into the same vertex. On the other hand, a vertex represents only one coordinate (s, t) in the parametric space, so that only one copy of the parametric coordinates should be persisted. This causes an obvious conflict in the data modeling work, as the ambiguity of vertex leads to its unclear definition. In some researches, the terms "vertex" and "control point" are even

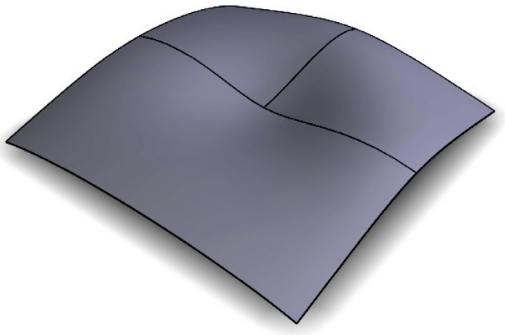


Fig. 4. A simple T-spline example for studying the new data models.

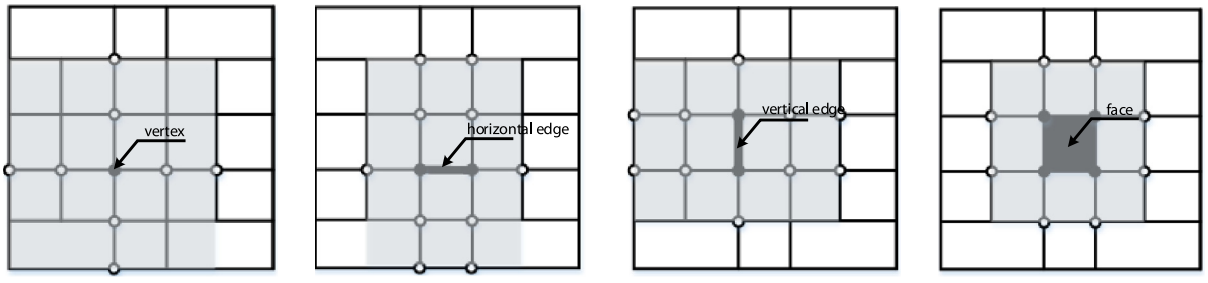
interchangeably used, although a vertex usually only represents a topological entity in the parametric space and a control point is the location of a vertex in the Cartesian space [15,23].

(3) Most current researches and applications focus on the T-spline of degree three. However, in case of even degrees the current principles of T-mesh may cause inconvenience in programming.

Unlike the regular control grid of B-spline, the existence of T-junctions in a T-mesh implies a significant problem in deducing knot vectors when the degree of a T-spline is even. As the T-spline is point-based (regarding the parametric aspect of a point, actually it should be called vertex-based), the knot space deducing principle may not work in some cases. For example, Fig. 3 presents a per-image of a T-mesh in different cases of even degrees in s and t directions. The current T-mesh researches must deduce the knot vectors from a vertex (Fig. 3(a)), while this inference works only when the T-mesh has odd degrees in both s and t directions. When one of the degrees is even, the deduction should be an either horizontal or vertical edge. When both of the degrees are even, the deduction should be from a face, as shown in Fig. 3((b)–(d)). In addition, multiplicity rules should be adjusted. Except for the vertex multiplicity, a T-mesh imposes the edge and face multiplicities as well. As a matter of fact, all the multiplicity definitions upon either a vertex, an edge, or a face cannot have universal functionalities.

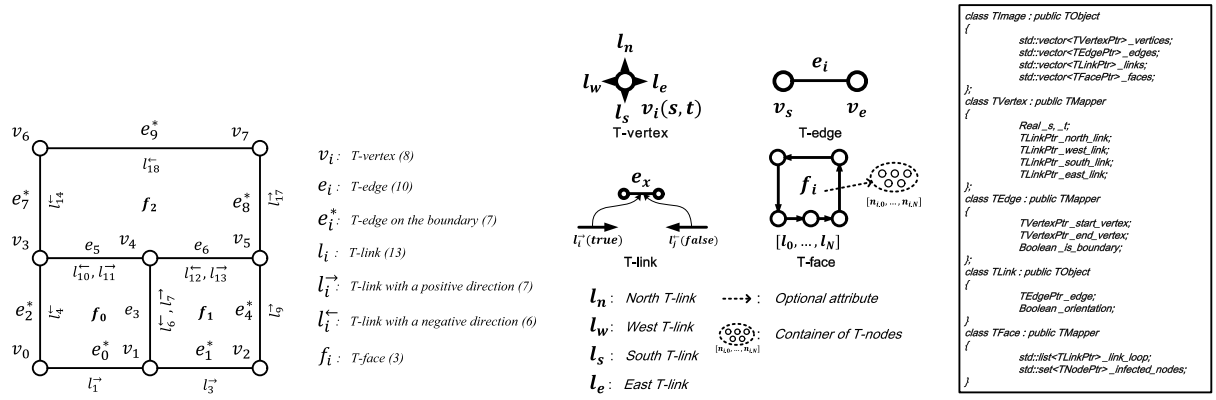
2.2. New T-spline data models

The main root of the aforementioned problems is the lack of effective data structures for processing and exchanging T-spline models. In order to solve this problem, this article redesigns the previous T-spline data models in a comprehensive manner. The proverbial T-mesh is decomposed into three-layer models, which represent the parametric, topological and Cartesian aspects of a T-mesh, respectively. This section analyzes the properties and internal relationships among them, and proposes a set of new data models correspondingly to avoid the ambiguousness. A simple T-spline surface as shown in Fig. 4 is studied as an example.



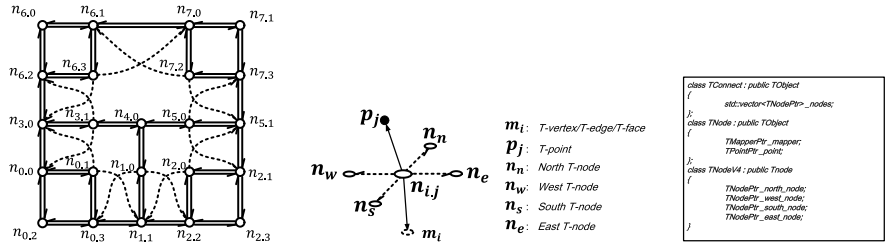
(a) Vertex based ($m = 3, n = 3$). (b) Edge based ($m = 2, n = 3$). (c) Edge based ($m = 3, n = 2$). (d) Face based ($m = 2, n = 2$).

Fig. 3. Knot space deducing principles in cases of different degrees (m, n) in s and t directions.



(a) A T-image example. (b) Data structures of parametric objects. (c) The C++ implementation.

Fig. 5. Data structures of the parametric objects and their C++ implementation.



(a) A T-connect example. (b) Data structures of topological objects. (c) The C++ implementation.

Fig. 6. Data structures of the topological objects and their C++ implementation.

2.2.1. Categories of information

(1) The parametric domain.

All the information that is drawable in the parametric space forms the parametric domain. According to this rule, the complicated knot multiplicity and knot space deduction are out of this scope, as they cannot be clearly "drawn" in the parametric space. Rest components include vertices (with a parametric coordinate), edges (with start and end vertices), links (with an edge and an orientation) and faces (bounded by a loop of links). In this paper, the parametric domain is referred to as the parametric layer.

(2) The connections between nodes.

The multiplicity and knot deduction are detached from the parametric domain, and simply unified as the connection information between different nodes, where a node contains only the topological connection of the formerly mentioned vertex, and two pointers to a vertex (or a edge, or a face) and a control point. It is worthwhile to note that a node hold neither parametric nor Cartesian coordinates, so there should be no distance between two nodes. Additionally, the former multiplicity happens when multiple nodes point to the same vertex (when degrees in both directions are odd), which does not break the consistent definition of a node. In this paper, the group of nodes is called the topological layer.

(3) The control points.

The control points represent the definitions of a T-mesh in the Cartesian space. A control point purely provides a (x, y, z) coordinate with a weight w and can be mapped one-to-one to a node without the consideration of multiplicities and connections. Comparing to the former T-mesh in the Cartesian space, the connections between different control points are subtracted, since this information has been previously illustrated in the topological layer. The control points constitute the Cartesian layer.

According to the above definitions, new data models of T-spline are introduced, which are organized into three layers correspondingly:

2.2.2. Parametric layer: T-image

A T-image contains the data models on the parametric layer, which means a set of T-vertices, T-edges, T-links and T-faces. A T-face is constituted by a loop of T-links. A T-link is actually an oriented edge, which refers to a T-edge and specifies its orientation using a Boolean. A T-edge is determined by its start and end T-vertices, while a T-vertex contains a parametric coordinate (s, t). The existence of the T-edge is to reduce the redundancy of T-links, as two T-links with opposite orientations may share the same T-edge. Fig. 5(a) gives an example of a T-image.

A T-image is in fact a dense graph, as the T-vertices, T-edges, T-links, and T-faces are all some fundamental elements of a standard directed graph. Fig. 5((b), (c)) shows the data structures of them. Note that a T-face has an optional container of T-nodes for the convenience of tessellation, which will be introduced in the next section.

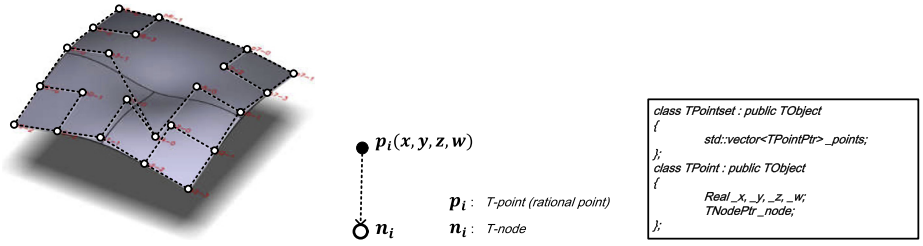
2.2.3. Topological layer: T-connect

A T-connect contains all the T-nodes which connect to each other. A T-node contains a pointer to a selection of a T-vertex, a T-link, or a T-face (according to the parity of degrees in s and t directions), another pointer to a control point(T-point), and a set of pointers to other T-nodes. The number of T-node pointers is called the valance of a T-node. Fig. 6(a) gives an example of a T-connect.

A T-connect is also a directed graph, but all the faces are empty and all connections have no distance value, so it is more like a grid without weights on its grid lines. Fig. 6((b), (c)) shows the data structures of the topological objects.

2.2.4. Cartesian layer: T-pointset

A T-pointset is a collection of T-points. A T-point is actually a rational point in the Cartesian space (with a weight) plus a pointer to a T-node, by which one can deduce the connection relationships between T-points. Fig. 7(a) presents a T-point group



(a) A T-pointset example. (b) Data structures of Cartesian objects. (c) The C++ implementation.

Fig. 7. Data structures of the Cartesian objects and their C++ implementation.

in the cartesian space, which is similar as the former plot of a T-mesh. Note that, a T-point group provides no connections between each other, though Fig. 7(a) deduces them from the corresponding T-connect and plots the results as dotted lines to enhance its visuality.

Generally, we call all the elements denoted by T as T-objects. The new data models of T-spline reflect the significant single-responsibility principle in terms of software engineering. Each layer of data models holds relatively consistent and stable information, and hence possesses homogeneous functionalities.

3. Characteristics of the new data models

According to the definition of the new models, the operations of T-spline become clear and interesting. New rules have to be imposed on, and new inferences can be deduced from the data structures.

3.1. T-image, T-connect and T-pointset

According to the definitions of T-image, T-connect and T-pointset, they are all similar to graphs, and none of them involve multiple vertices. This implies the classical graph theory [24] could be easily introduced for analyzing and manipulating a T-spline. Fig. 11 shows the linkages among a T-image, a T-connect and a T-pointset.

A T-image is a direct graph with the most abundant T-objects, including T-vertices, T-edges, T-links, and T-faces, which are usually the most complex data in a T-mesh. Each T-edge in a T-image has a distance and a direction, and a T-vertex does not directly connect to its adjacent T-vertices but T-links. For a T-vertex with its valance equal to 4, there are notations "north", "west", "south" and "east" on each T-link pointers of a T-vertex.

A T-connect contains only the topological connections, and is also a directed graph. However, there is no concept of edges and faces inside a T-connect. Each T-node connects directly to its neighbors, and the connection has no distance. The T-node also has notations of "north", "west", "south" and "east" on its connections.

A T-pointset is simply constituted by a set of T-points. Each T-point is in fact a rational point, which contains not only a (x, y, z) coordinate but also a weight w. No connections exist between T-points, so a T-pointset usually provides the least information.

Basically, there are two one-to-one inferences imposed on the three-layer models:

Inference 1: The patches on a T-spline surface one-to-one map to the T-faces on a T-image.

Inference 2: The T-points in a T-pointset one-to-one link to the T-nodes (Non-virtual) in a T-connect.

3.2. T-junctions and virtual T-objects

There are two kinds of T-junctions in the parametric and topological layers, respectively. For the parametric layer, a T-junction exists when a T-vertex has one and only one empty pointer to its adjacent T-links. Since each pointed T-link has a notation of direction, this T-junction also has a similar notation, which is specified by the missed pointer. For the topological layer, a T-junction means a T-node that misses one and only one adjacent T-node. The notation of this T-junction is also specified by the missed direction. Fig. 8 shows the directions of T-junction.

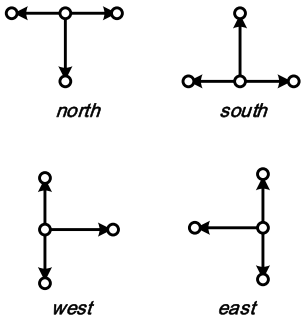


Fig. 8. Directions of a T-junction.

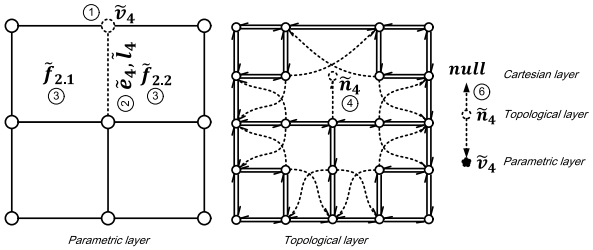
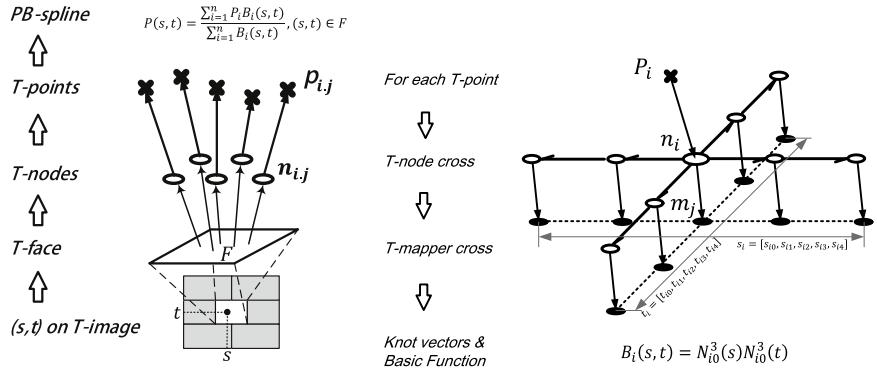


Fig. 9. Virtual T-objects.

When a T-junction occurs, the patching operation needs to be implemented, so that a complete pair of knot vectors can be inferred for calculating the related Bernstein basis function. Virtual T-objects are some auxiliary elements that help for the calculation purposes, and can be generated automatically in run time. Virtual T-objects include virtual T-nodes, virtual T-vertices, virtual T-links, virtual T-edges and virtual T-faces. Fig. 9 presents the deducing process of virtual T-objects.

3.3. Dual indexing roles of a T-node

Comparing to a T-vertex, a T-node is relatively simpler, as it represents only the connections between each other. In spite of that, a T-node may play two indexing roles in the T-spline models. On one hand, the upwards indexing role is played to trace all T-points from a T-face, as a non-virtual T-node has an one-to-one mapping relationship to a T-point. On the other hand, the downwards indexing role is played for inferring the knot vectors when the point-based blending function needs that for computation. Fig. 10 demonstrates the principle of the dual



(a) Upwards indexing procedure. (b) Downwards indexing procedure.

Fig. 10. Dual indexing roles of T-nodes played in the upwards and downwards indexing procedures.

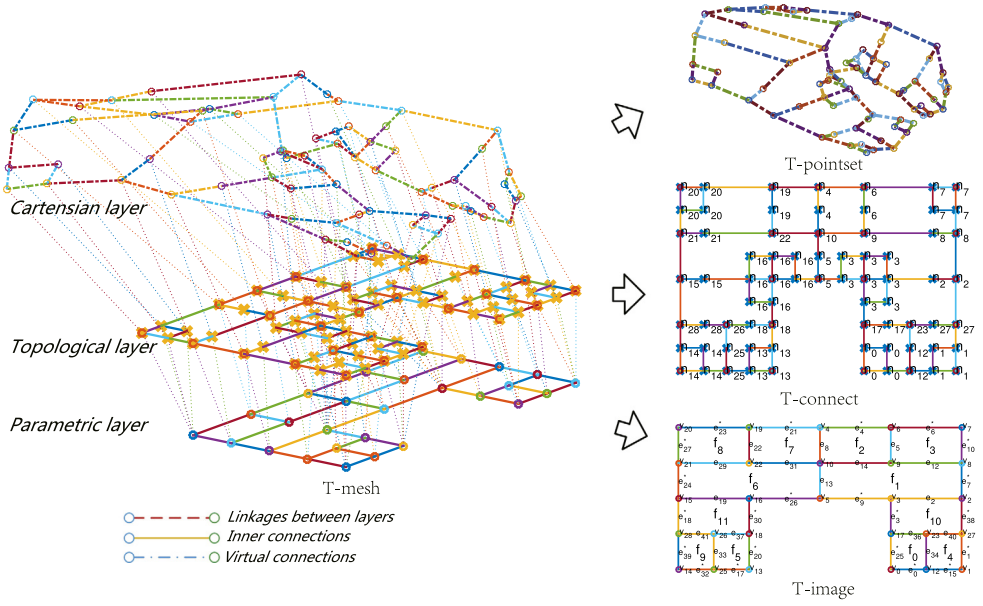


Fig. 11. Linkages among a T-image, a T-connect and a T-pointset.

indexing roles. Correspondingly, the path from a parametric coordinate (s, t) to a T-point is called the upwards indexing procedure and the path from a T-point to the knot vectors is called the downwards indexing procedure. The dual roles of a T-node usually coexist, while there are also exceptions. When a virtual T-node is employed to the blending function, typically it has an empty pointer to T-point, so only the downwards indexing role exists in this case.

3.4. Efficient data accessing mechanisms

Since the new data models are different from the pre-image of T-mesh, their data accessing and manipulating mechanisms are also different. Basically, any T-objects can be used as the entry of data accessing, and all other wanted data can be traceable from any start points. For example, start from a T-point, its parametric domain used for the tensor product of the two B-spline curves can be obtained using an efficient data accessing mechanism ($m = n = 3$):

1. Find the T-node Since each T-point has a pointer to a T-node, the corresponding T-node can be obtained directly;

2. Find the T-node cross A T-node cross consists of a center T-node and for each orientation (degree -1) linked T-nodes, as shown in Fig. 10(b). Each T-node has four pointers to its adjacent north, west, south and east T-nodes, so the T-node cross can be obtained by iteratively inferring the adjacent T-node in individual directions.

3. Find the T-vertex cross Similarly, a T-vertex cross contains a center T-vertex and four lists of T-vertices. Whereas, the number of T-vertices on each direction does not constantly equal to (degree -1), because multiple T-nodes may point to the same T-vertex. In spite of that, there is no need to impose any special operations on the finding procedure.

4. Find two knot vectors The T-vertex cross contains the parametric coordinations. Hence, the two knot vectors can be deduced directly.

Each step of the deduction of the knot vectors is direct data accessing operations, and has no need to prestore the redundant knot values.

3.5. Degree parity and generalized multiplicity

Most researches on T-spline have limited their scopes on cubic (3 degrees) T-splines [1,23,25]. In this case, a T-point(T-node) can be somehow mapped to a T-vertex directly, so the pre-image of T-mesh can reluctantly represent the parametric domain of a

T-spline surface and the knot vectors can be pre-stored in vertices. However, this conclusion can only be formed when the degree is odd. When the even degree occurs, a T-point(T-node) may not map to a T-vertex but to a T-edge or a T-face in many cases. Some rules can be derived from the existence of the generalized multiplicity:

Rule 1 If a T-spline has odd degrees in both s and t directions, a T-node should only point to a T-vertex.

Rule 2 If a T-spline has an odd degree in the s direction and an even degree in the t direction, a T-node should point to either a T-vertex or a vertical T-edge. When a T-node points to a T-vertex, it means there are two multiple knots in the t direction.

Rule 3 If a T-spline has an even degree in the s direction and an odd degree in the t direction, a T-node should point to either a T-vertex or a horizontal T-edge. When a T-node points to a T-vertex, it means there are two multiple knots in the s direction.

Rule 4 If a T-spline has even degrees in both s and t directions, a T-node should point to either a T-vertex, a T-edge (either vertical or horizontal), or a T-face. When a T-node points to a vertical T-edge, it means there are two multiple knots in the s direction. When a T-node points to a horizontal T-edge, it means there are two multiple knots in the t direction. When a T-node points to a T-vertex, it means there are two multiple knots in both s and t directions.

3.6. Data and pointer redundancy

In the software engineering, it is a commonly used strategy to store some redundant data or pointers to save in-time computations, so as to fasten the run time speed. According to the implement-in-terms-of rule [26], redundant pointers are much more efficient than redundant data. Hence, redundant data are strictly avoided in the new T-spline data models. Whereas, some redundant pointers are still necessary to decrease the time complexity. These redundant pointers can be obtained either from exchanged data or initialization when the data are loaded. One typical example is the set of infected T-nodes in a T-face. From Eq. (1) and Fig. 10, we can derive two essential inferences:

Inference 3: Each T-point should have its individual knot vectors.

Inference 4: Each T-face should have its individual set of T-nodes (called infected T-nodes).

Inf. 3 can be guaranteed using the downwards indexing procedure, while Inf. 4 means a set of T-nodes should be pre-assigned to each T-face. This operation can be carried out during initialization. Fig. 12 presents the initializing flowchart of the sets of infected T-nodes in all T-faces.

3.7. Separated patch tessellation

The tessellation is a key functionality for representing a spline surface. As the T-faces have one-to-one mapping relationships with the patches of a T-spline surface, the tessellation process can be separated according to T-faces. After initialization, each T-face has its crucial data for computation (equivalent to a PB-spline [1]), hence a relatively high efficiency can be achieved during tessellation. Furthermore, the patch oriented tessellation principle facilitates to adjust the face based resolution, so that small patches may have finer meshes. Since the T-faces have all the required data, the separated patch tessellation brings no iterative computations to split the parametric domain. This adjustment significantly balances the tessellating performance and quality. On one side, large patches need to be tessellated more coarsely to save the computation time. On the other side, small patches have to be tessellated more finely, so that details may not be overlooked. Fig. 13 gives an example of the face based tessellation.

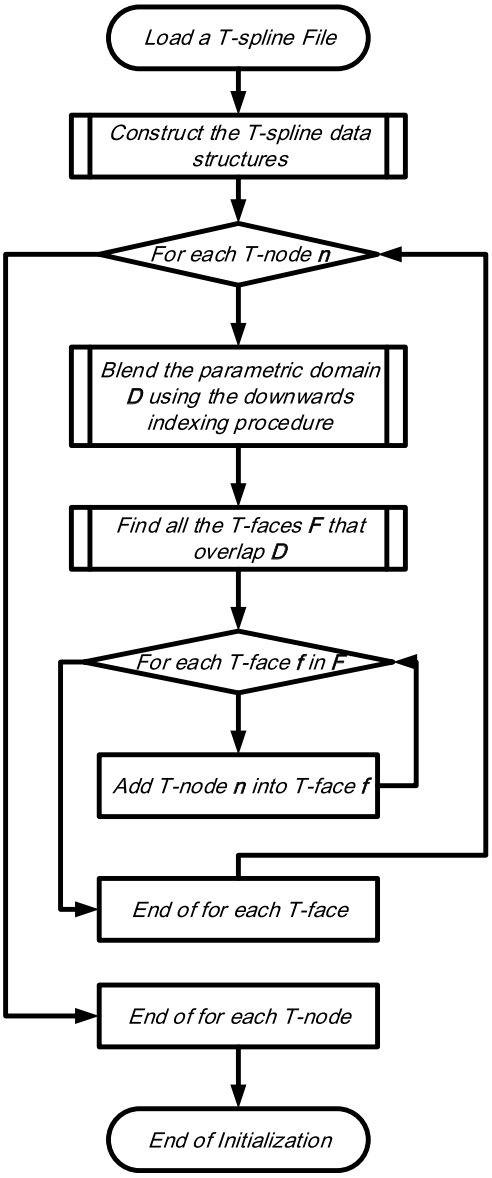


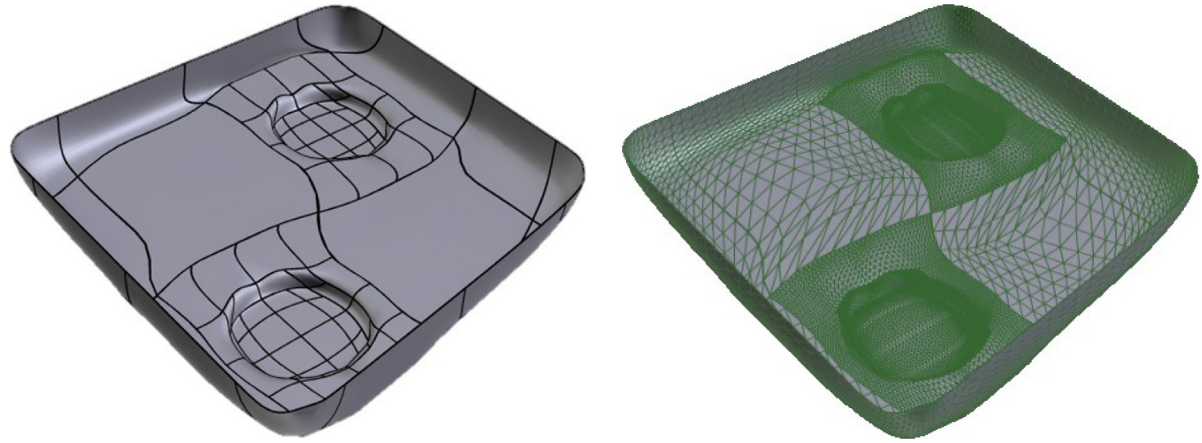
Fig. 12. Initializing the sets of infected T-nodes in all T-faces.

3.8. Stable data storage against modifications

The proposed data models have great advantages against modifications. According to the definition, the separation of T-image, T-connect and T-pointset also distinguishes the modification frequencies accordingly. The most frequent changes happen on the Cartesian layer, the topological layer suffers less, and the parametric layer is the most stable data structure in memory. When a control point changes its position, only the Cartesian layer needs to be adjusted. If a control point changes its neighbors or vanishes, the topological layer also needs to be changed. Only if the topological structure of a T-spline surface is transformed, which means only when a quantity-to-quality change happens, the parametric layer needs to be touched. A relatively stable data storage makes it more convenient to manipulate the T-spline data structures and more efficient to handle with modifications and adjustments, especially in cases of large T-spline models.

4. STEP-compliant T-spline data models

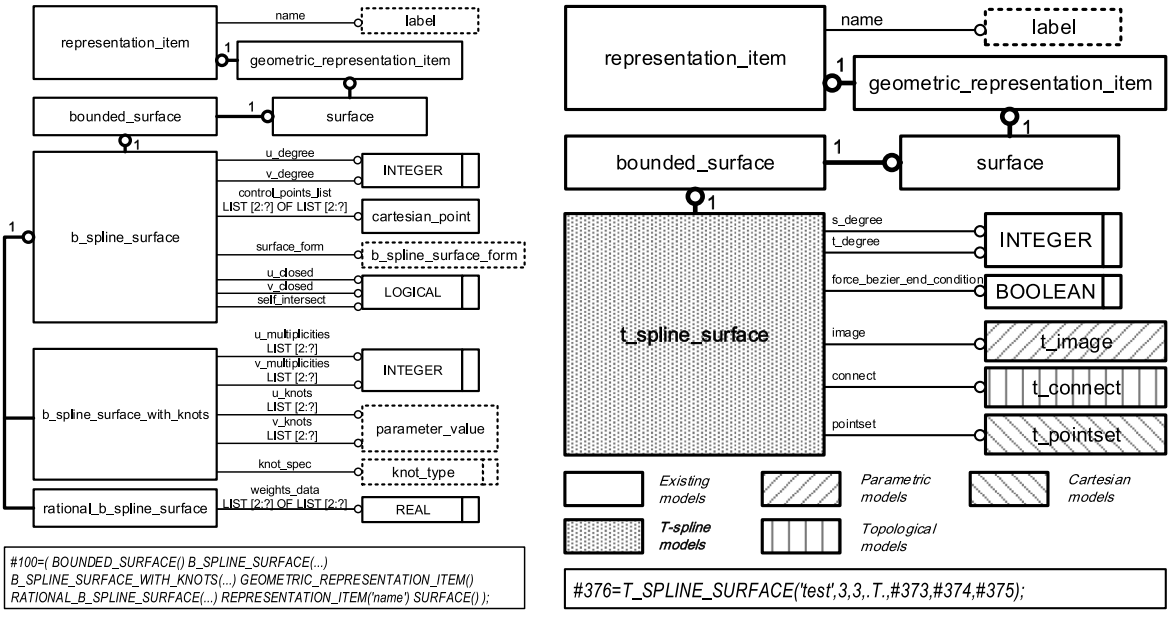
This section applies the new T-spline data models using the EXPRESS language [18], which can inherently merge the



(a) A T-spline surface with different patch sizes.

(b) The face based tessellation result.

Fig. 13. An T-spline example of the face based tessellation.



(a) The STEP definitions of a NURBS surface.

(b) The STEP definitions of a T-spline surface.

Fig. 14. Comparing the STEP definitions of NURBS and T-spline surfaces.

object-oriented architecture into the T-spline data structures. Newly defined T-spline models are embedded into the AP238 standard (STEP-NC AIM) [27], in which a lot of modeling work can be saved by using inheritances and compositions of the abundant existing entities and types. Amount of EXPRESS-G diagrams are illustrated to present the hierarchical structures. According to the previous definitions, STEP-compliant T-spline models are separated into three groups: the parametric, topological and Cartesian models. Before the introduction and discussion of the new STEP models, the NURBS models that are defined in the recent STEP standard are reviewed for the comparison with T-spline models.

4.1. Comparison of NURBS and T-spline models

In the recent STEP standards, NURBS is the only mathematical model to describe a freeform surface. Comparing T-spline, NURBS is significantly simpler. Fig. 14(a) illustrates the EXPRESS definition of NURBS in the STEP standard [20,21]. A NURBS surface is defined in a compounded entity, where the mathematical model is split into three major entities: b_spline_surface, b_spline_surface_with_knots and rational_b_spline_surface. Obviously, the EXPRESS models of a NURBS surface are designed to be grid based, for the knot vectors with their multiplicities are defined using arrays ('u_knots', 'v_knots', 'u_multiplicities' and 'v_multiplicities'). Due to the aforementioned reasons, these models cannot be directly extended to the use of a T-spline surface. Therefore, T-spline EXPRESS models have to be designed solely. Fig. 14(b) shows the STEP definitions of a T-spline surface. Basically, a T-spline surface is determined by a t_image, a t_connect and a t_pointset, whose definitions will be introduced detailedly in the following parts of this section.

4.2. Parametric models

The parametric models include the entities and types t_vertex, t_edge, t_link, t_edge_condition, t_face, t_mapper, t_image, etc. The entity t_vertex represents the model of a T-vertex, the entity t_edge stands for a T-edge, and others are obviously understood in a similar way. The EXPRESS-G diagram of parametric models are presented in Fig. 15.

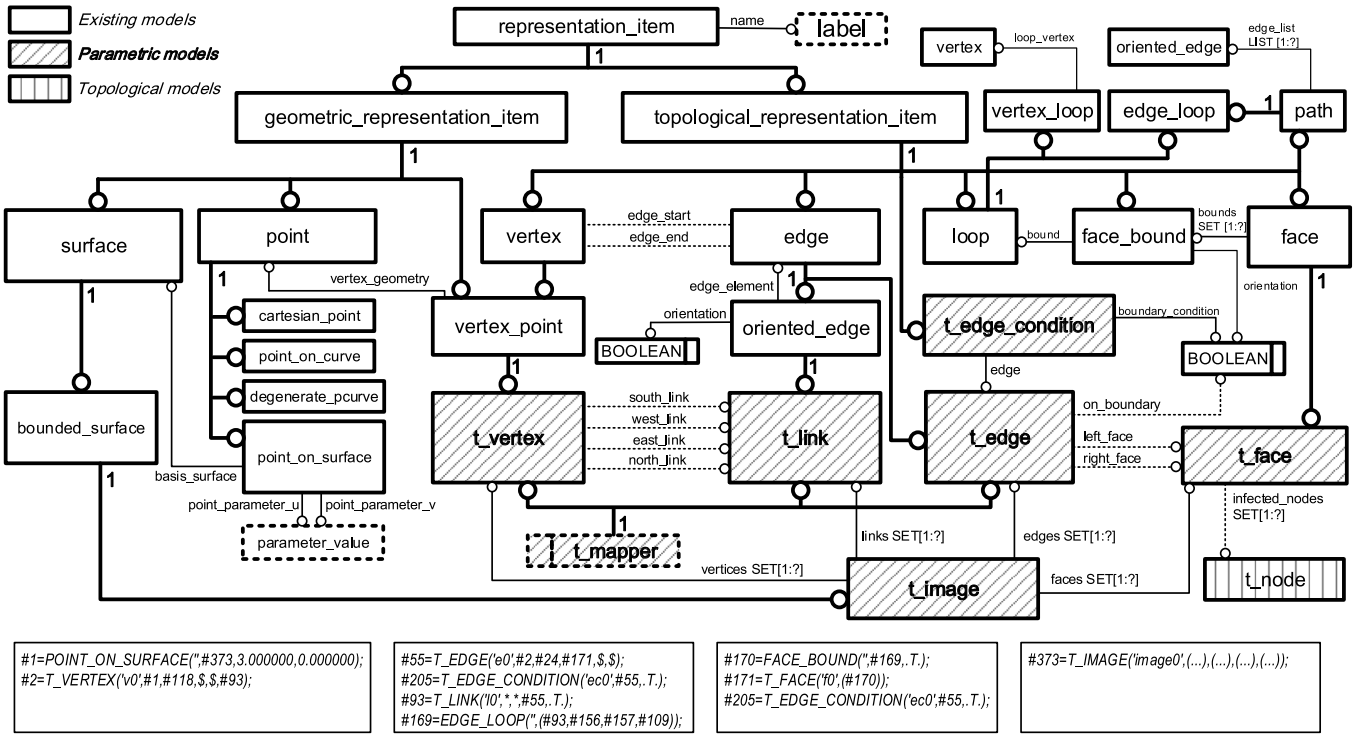


Fig. 15. EXPRESS diagrams of the parametric models.

The entity t_vertex inherits from a vertex_point, whose hierarchical structures are given in the existing standard. A vertex_point basically inherits from two entities: vertex and geometric_representation_item, and contains an attribute vertex_geometry of a point type. The entity point is also given in the existing STEP standard, which is the supertype of different types of points, such as cartesian_point, point_on_curve, point_on_surface, degenerate_pcurve, etc. Wherein, the entity point_on_surface is introduced to T-spline for the definition of a parametric coordinate. The entity point_on_surface inherits from the entity point, and contains an attribute basis_surface of type surface which holds the target parametric domain (in this case, a t_image), and attributes point_parameter_u and point_parameter_v of type parameter_value to store the (u, v) coordinate. The entity t_edge is a subtype of the entity edge, which provides the attributes edge_start and edge_end of type vertex for the reference to t_vertex(t_vertex inherits from vertex). The entity t_link inherits from the entity oriented_edge, which gives a Boolean attribute to specify the orientation. The entity t_edge_condition is an auxiliary entity to the entity t_edge. When the optional Boolean attribute on_boundary of a t_edge is not assigned, a t_edge_condition entity can be used as a substitution. This method helps to reduce some redundant data in a large T-spline file. The entity t_face is a subtype of the entity face containing a set of entities face_bound. The entity face_bound is defined by a bound of type loop and a Boolean orientation, while the entity loop derives the entities vertex_loop and edge_loop. The entity edge_loop has another supertype of the entity path, in which an edge list of type oriented_edge (the parent of t_link) is provided to determine the boundary of the t_face. Additionally, the entity t_face uses an optional attribute infected_nodes to store the required T-nodes in the upwards indexing procedure (Fig. 10). The entity t_image is derived from the entity bounded_surface, and is composed of sets of t_face, t_edge, t_link and t_vertex entities. The type t_mapper is a select of a t_vertex, a t_edge, or a t_face, all of which can be used as a T-mapper on the parametric layer.

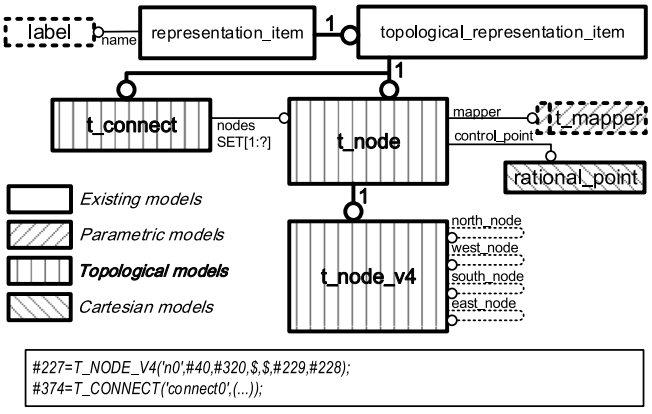


Fig. 16. EXPRESS diagrams of the topological models.

4.3. Topological models

Topological models consist of only two types of entities, t_node and t_connect.

A t_node is an abstract supertype of all other t_node entities. As presented in Fig. 16, the entity t_node has an attribute vertex of type t_vertex and an attribute control_point of type rational_point. A t_node gives no information how T-nodes connect each other, as these definitions should be node type dependent. In this paper, the entity t_node_v4, which inherits from the entity t_node, defines the connections to its four t_node_v4 neighbors via four optional pointers (north_node, west_node, south_node, and east_node). The entity t_connect possesses a set container attribute to organize all the t_nodes involved on the topological layer.

4.4. Cartesian models

Cartesian models represent the Cartesian layer of T-spline, and contain only two entities, rational_point and t_pointset. The entity

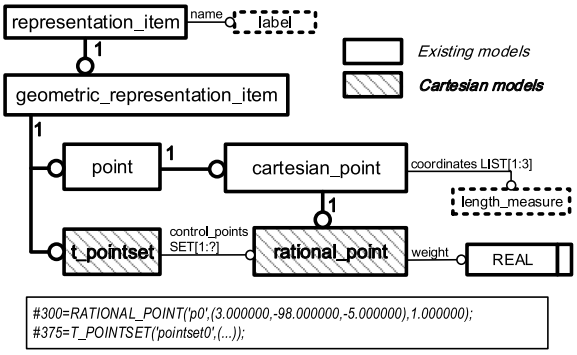


Fig. 17. EXPRESS diagrams of the Cartesian models.

rational_point inherits from a cartesian_point, so it essentially possesses the (x, y, z) coordinate. The notation "rational" means it additionally sets a weight value on each point. The entity t_pointset is simply a container of all rational_points involved on the Cartesian layer (see Fig. 17).

5. Prototype system

This paper developed a prototype system to verify the newly designed T-spline models which consists of three major components: the TSM convertor, the STEP parser, and the T-spline kernel. Fig. 18 shows the prototype system to test the STEP-compliant data exchange of T-spline, and Fig. 19 gives some testing results.

(1) TSM convertor: convert a TSM file to a STEP file.

The recent most frequently used T-spline modeling tool is the Rhino software with a T-spline plug-in contributed by the T-splines company. While, the clear text file format exported by the Rhino system is called TSM file. In order to obtain T-spline models from Rhino, a TSM to STEP convertor is developed. As this work is a temporary use, the convertor is developed using the Matlab system, which can save a lot of programming efforts.

(2) STEP parser: read a STEP file into memory.

Once the STEP file is obtained, they should be firstly read into the memory and parsed clearly. In order to fulfill this demand, a standard procedure called SDAI (Standard Data Access Interface) has to be implemented. All EXPRESS models embedded inside the AP238 were automatically converted into C++ language, and so can be compiled to an executable library.

(3) T-spline kernel: manipulate the T-spline models.

The exchanged T-spline model finally has to be handled by a software kernel. This paper developed a T-spline kernel using the architecture constituted by the aforementioned models. This T-spline kernel realizes the redesigned T-spline models and supports the related algorithms.

Using STEP-compliant T-spline models, it will increase the size of data files for storage, Table 2 presents the comparison between TSM and STEP files. Although it increases the storage cost in STEP-compliant T-spline files (this is actually an issue for all STEP files), the cost is definitely worth comparing to the brought convenience and legibility.

Table 2
Sizes of TSM and STEP files.

Models	TSM (kB)	STEP (kB)	Ratio
Simple	1.763	5.624	3.19
Mouse	6.277	19.239	3.06
Blade surface	16.244	30.325	1.87
Gearbox cover	28.362	76.058	2.68
Bike seat	45.083	74.910	1.66
Human face	66.698	116.651	1.75

6. Discussion and conclusion

This paper reconsiders the drawbacks of the existing T-spline models, and proposes a set of new T-spline data models to obtain better data storing and operating efficiencies. The conventional T-mesh is decomposed into the parametric, topological, and Cartesian layers, respectively. The characteristics of the redesigned T-spline models are analyzed and reviewed. Using the proposed T-spline data structures, the STEP-compliant data models are presented for the standardized data exchange between different CAx systems in the future. A prototype system which consists of a TSM-to-STEP convertor, a STEP parser and a T-spline kernel, is proposed to implement the data models and realize the

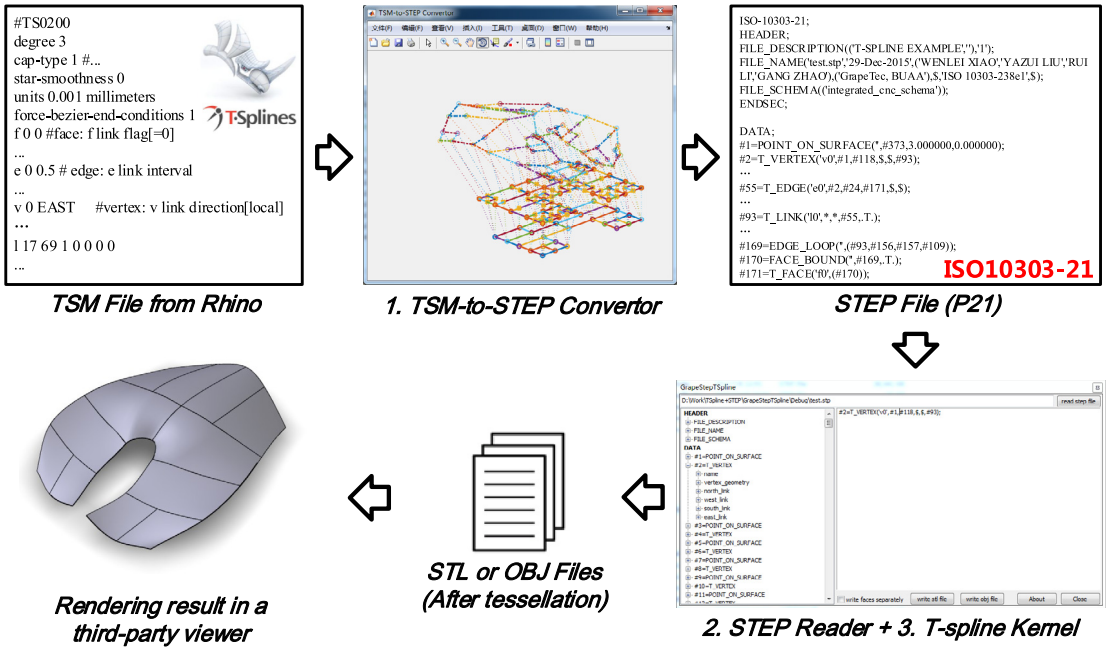
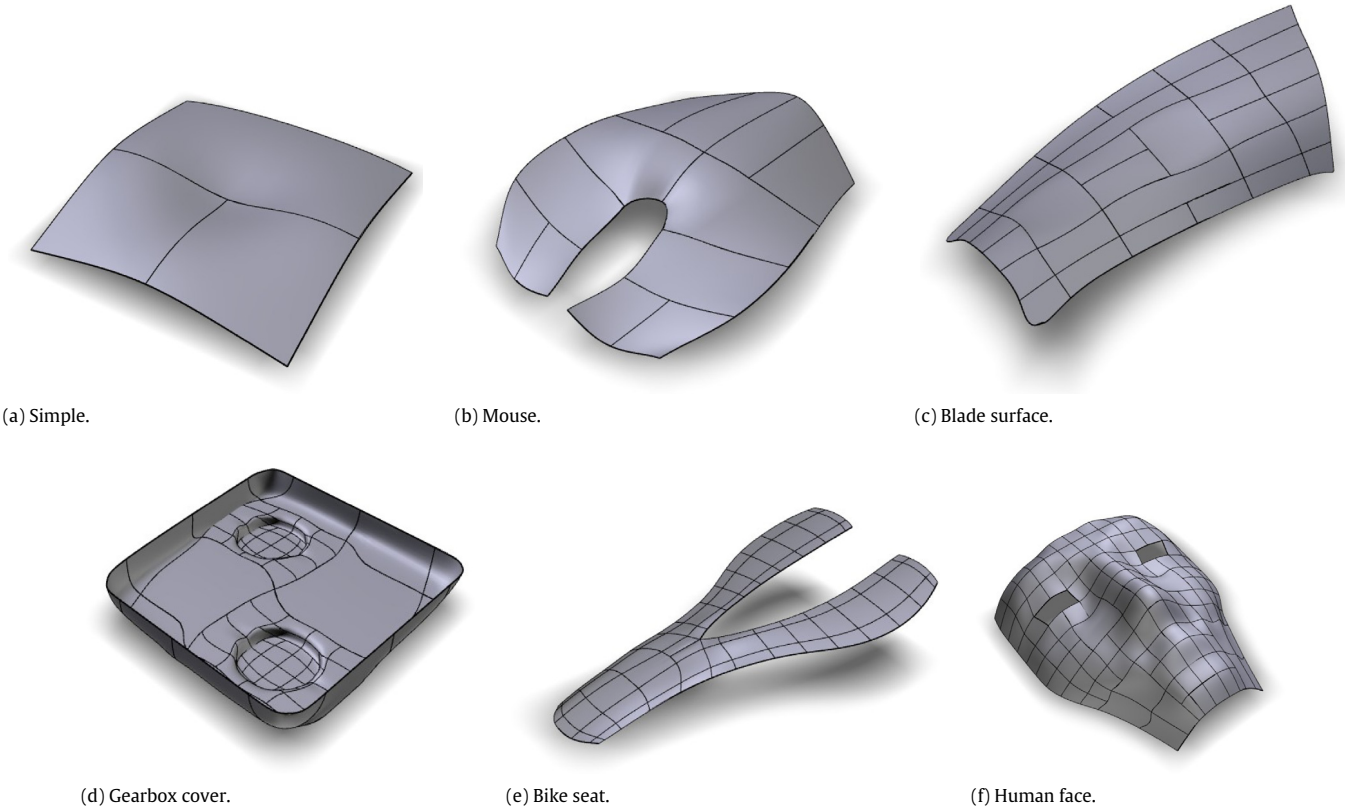


Fig. 18. A prototype system to test the STEP-compliant data exchange of T-spline models.



(a) Simple. (b) Mouse. (c) Blade surface.

(d) Gearbox cover. (e) Bike seat. (f) Human face.

Fig. 19. STEP-compliant data exchange testing results.

STEP-compliant data exchange. In order to verify the feasibility and efficiency, some T-spline examples have been tested and presented. The testing results prove that the new T-spline data models are much more friendly to both human and computer, hence have a great prospect for the future development of T-spline technologies. Although recent data models work well for the regular T-spline surfaces, other more generalized models have to be considered in the future. For example, unstructured T-splines with extraordinary vertices should be considered [28], T-NURCCS models will be supported for a manifold T-spline surface [1,29], volumetric T-spline will be defined for models with higher dimensions [30,31], and the research and development of new-model based algorithms should be investigated and discussed.

Acknowledgments

This research was financially supported by the National Natural Science Foundation of China (No. 51505020, No. 61572056, and No. 31327901). We acknowledge to anonymous reviewers for their comments and advice.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.cad.2016.06.004.

References

[1] Sederberg TW, Zheng J, Bakenov A, Nasri A. T-splines and T-NURCCs. ACM Trans Graph 2003;22(3):477–84. http://dx.doi.org/10.1145/882262.882295.

[2] Sederberg TW, Cardon DL, Finnigan GT, North NS, Zheng J, Lyche T. T-spline simplification and local refinement. ACM Trans Graph 2004;23(3):276–83. http://dx.doi.org/10.1145/1015706.1015715.

[3] Zheng J, Wang Y, Seah HS. Adaptive T-spline surface fitting to z-map models. In: Proceedings of the 3rd international conference on computer graphics and interactive techniques in australasia and south east Asia. GRAPHITE'05, New York (NY, USA): ACM; 2005. p. 405–11. http://dx.doi.org/10.1145/1101389.1101468.

[4] Xu G, Mourrain B, Duvigneau R, Galligo A. Analysis-suitable volume parameterization of multi-block computational domain in isogeometric applications. Comput-Aided Des 2013;45(2):395–404. http://dx.doi.org/10.1016/j.cad.2012.10.022.

[5] Xu G, Mourrain B, Duvigneau R, Galligo A. Optimal analysis-aware parameterization of computational domain in 3D isogeometric analysis. Comput-Aided Des 2013;45(4):812–21. http://dx.doi.org/10.1016/j.cad.2011.05.007.

[6] Autodesk. T-Splines Plug-in for Rhino, 2014. http://www.autodesk.com/education/free-software/t-splines-plug-in-for-rhino.

[7] Bazilevs Y, Calo V, Cottrell J, Evans J, Hughes T, Lipton S, et al. Isogeometric analysis using T-splines. Comput Methods Appl Mech Engrg 2010;199(58):229–63. http://dx.doi.org/10.1016/j.cma.2009.02.036.

[8] Ginnis A, Kostas K, Politis C, Kaklis P, Belibassakis K, Gerostathis T, et al. Isogeometric boundary-element analysis for the wave-resistance problem using T-splines. Comput Methods Appl Mech Engrg 2014;279:425–39. http://dx.doi.org/10.1016/j.cma.2014.07.001.

[9] Gan WF, Fu JZ, Shen HY, Chen ZY, Lin ZW. Five-axis tool path generation in CNC machining of T-spline surfaces. Comput-Aided Des 2014;52:51–63. http://dx.doi.org/10.1016/j.cad.2014.02.013.

[10] Lai J, Fu J, Shen H, Gan W, Chen Z. Machining error inspection of T-spline surface by on-machine measurement. Int J Precis Eng Manuf 2015;16(3):433–9. http://dx.doi.org/10.1007/s12541-015-0059-4.

[11] Newman S, Allen Jr R, RR. CAD/CAM solutions for STEP-compliant CNC manufacture. Int J Comput Integr Manuf 2003;16(7–8):590–7. http://dx.doi.org/10.1080/0951192031000115688.

[12] Xu X. Realization of STEP-NC enabled machining. Robot Comput-Integr Manuf 2006;22(2):144–53. http://dx.doi.org/10.1016/j.rcim.2005.02.009.

[13] Xiao W, Zheng L, Huan J, Lei P. A complete CAD/CAM/CNC solution for STEP-compliant manufacturing. Robot Comput-Integr Manuf 2015;31:1–10. http://dx.doi.org/10.1016/j.rcim.2014.06.003.

[14] T-Splines Inc.. TSM (T-spline Mesh) file format, 2015. www.tsplines.com.

[15] Asche C, Berkhahn V. Efficient data structures for T-spline modeling. In: 19th international workshop of the European group for intelligent computing in engineering. Herrsching, Germany. 2012.

[16] CGAL. The Computational Geometry Algorithms Library, 2015. http://www.cgal.org.

[17] Lin H, Cai Y, Gao S. Extended T-mesh and data structure for the easy computation of T-spline. J Inf Comput Sci 2012;9(3):583–93.
[18] ISO 10303-11. Industrial automation systems and integration – product data representation and exchange – part 11: Description methods: The express language reference manual. 2004.
[19] ISO 10303-23. Industrial automation systems and integration – product data representation and exchange – part 23: Implementation methods: C++ language binding to the standard data access interface. 2000.
[20] ISO 10303-203. Industrial automation systems and integration – product data representation and exchange – part 203: Application protocol: Configuration controlled 3d design of mechanical parts and assemblies. 2011.
[21] ISO 10303-214. Industrial automation systems and integration – product data representation and exchange – part 214: Application protocol: Core data for automotive mechanical design processes. 2003.
[22] Scottc M, Hughesa T, Sederbergb T, Sederbergd M. An integrated approach to engineering design and analysis using the Autodesk T-spline plugin for Rhino3d. 2013.
[23] Scott M, Li X, Sederberg T, Hughes T. Local refinement of analysis-suitable T-splines. Comput Methods Appl Mech Engrg 2012;213216:206–22. http://dx.doi.org/10.1016/j.cma.2011.11.022.
[24] Bondy JA. Graph theory with applications. Oxford (UK, UK): Elsevier Science Ltd.; 1976.
[25] Cardon DL. T-spline simplification (Master of science), Brigham Young University; 2007.
[26] Meyers S. Effective C++. Addison Wesley; 2005.
[27] ISO 10303-238. Industrial automation systems and integration - product data representation and exchange - part 238: Application protocol: Application interpreted model for computerized numerical controllers. 2008.
[28] Scott MA, Simpson RN, Evans JA, Lipton S, Bordas SPA, Hughes TJR, et al. Isogeometric boundary element using unstructured T-splines. Comput Methods Appl Mech Engrg 2013;254. http://dx.doi.org/10.1016/j.cma.2012.11.001.
[29] Gu X, He Y, Qin H. Manifold splines. Graph Models 2006;68(3):237–54. http://dx.doi.org/10.1016/j.gmod.2006.03.004.
[30] Wang K, Li X, Li B, Xu H, Qin H. Restricted trivariate polycube splines for volumetric data modeling. IEEE Trans Vis Comput Graphics 2012;18(5):703–16. http://dx.doi.org/10.1109/TVCG.2011.102.
[31] Zhang Y, Wang W, Hughes TJ. Conformal solid T-spline construction from boundary T-spline representations. Comput Mech 2013;51(6):1051–9. http://dx.doi.org/10.1007/s00466-012-0787-6.