

CONVERSION BETWEEN T-SPLINES AND HIERARCHICAL B-SPLINES

Yimin Wang Jianmin Zheng Hock Soon Seah

School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798
email: {wang0066, asjmzheng, ashssseah}@ntu.edu.sg

ABSTRACT

T-splines is a recently developed surface modelling technique which is a generalization of B-splines and allows true local refinement. Another well established method supporting local refinement is hierarchical B-splines. This paper presents algorithms for transformation between T-splines and hierarchical (rational) B-splines. With the transformation, a surface expressed in terms of T-splines can be converted into a hierarchical (rational) B-spline representation and vice versa. The correspondence between T-spline representation and hierarchical (rational) B-spline representation is generally not one-to-one. Our conversion algorithms can yield compact representations.

KEY WORDS

Surface modelling, surface representation, hierarchical B-splines, T-splines, local refinement.

1 Introduction

This paper addresses the issue of conversion back and forth of a surface between the T-spline and hierarchical (rational) B-spline representations. T-splines [1] and hierarchical B-splines [2] are two different generalizations of B-spline surfaces. Both allow the surface to be refined locally.

B-splines have been widely used as a fundamental representation for freeform shapes in geometric modelling and computer animation systems [3]. In a three-dimensional modelling environment, there is quest for locally editing and modifying the fine scale details of the shape. For a B-spline curve, the finer control for details can be gained through knot insertion which is a local refinement process and restricts the influence of the control point to a local area. For a tensor-product B-spline surface, however, knot insertion is not a local process because the insertion of one knot into the surface knot vectors causes creation of new control points along an entire row or column. To overcome this drawback, Forsey and Bartels [2] introduced the concept of hierarchical B-splines. The hierarchy of surface refinements provides the capability for local refinement of surfaces and multiresolution surface editing. Hierarchical B-splines have been successfully used in Forsey's interactive modeller called "Dragon" and facial animation. A recent progress in enhancing local refinement for B-spline surfaces is T-splines which is a new modelling technology allowing local refinement [1]. T-splines generalizes B-splines by permitting T-junction points in their

control grid. T-splines are capable of eliminating superfluous control points from the B-spline models.

The main difference between T-splines and hierarchical B-splines is really between T-junctions and a hierarchy. With hierarchical B-splines, one could have a hierarchy which can be desirable for some applications, especially for multiresolution modelling. With T-splines, one could work on a single layer mesh, which contains substantially fewer points compared to the B-spline control mesh. Both representations have their own strength. Therefore it could be useful in practice to be able to convert one representation into another. The purpose of this paper is thus to provide such conversion algorithms. The previous work has shown that a hierarchical (rational) B-spline surface can be converted into a (rational) B-spline surface with finer control mesh, and so does a T-spline surface. There are also algorithms for constructing hierarchical B-spline surfaces [4] and T-spline surfaces [5] to approximate a given B-spline surface. If we set the approximate error to be zero, then the approximate algorithms will lead to exact conversion from B-splines to hierarchical B-splines or T-splines. Using these algorithms, one can achieve the conversion between hierarchical (rational) B-splines and T-splines through the *underlying* (rational) B-splines. However, this is an indirect approach. It does not take into consideration the structures of hierarchical B-splines or T-splines. In this paper, we present direct conversion algorithms. The algorithms will make heavy use of the local vertex insertion algorithms developed in [5].

The paper is organized as follows. Section 2 provides the necessary background on T-splines and hierarchical B-splines. Section 3 presents the algorithm for converting a hierarchical (rational) B-spline surface to a T-spline surface. In Section 4 follows the algorithm for converting a T-spline surface to a hierarchical (rational) B-spline surface. Section 5 concludes with discussion. We limit our discussion to the case of degree three B-splines and T-splines with only single knots for simplicity.

2 T-splines and hierarchical B-splines

This section gives a brief review of T-splines and hierarchical B-splines. Considering that a T-spline surface is generally a rational surface, we also generalize the concept of hierarchical B-splines to the rational case.

2.1 T-splines

T-splines are non-uniform rational B-spline (NURBS) surfaces with T-junctions. The control grid for a T-spline surface is called a T-mesh. The T-mesh is similar to the NURBS control mesh except that in a T-mesh a row or column of control points is permitted to terminate. The final control point in a partial row or column is a T-junction. The T-junctions enable T-spline surfaces to be refined locally. That is, it is possible to add a single control point to a T-spline control grid without propagating an entire row or column of control points and without altering the surface. Knot information for a T-spline is expressed using knot intervals that indicate the difference between two knots and are assigned to the edges in the T-mesh. The equation for a T-spline surface in homogeneous form is

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

where the $P_i = (w_i x_i, w_i y_i, w_i z_i, w_i)$ are control points in 4D space and the w_i are control point weights. The $B_i(s, t)$ are T-spline basis functions corresponding to control point P_i and given by

$$B_i(s, t) = N_i(s)N_i(t)$$

where $N_i(s), N_i(t)$ are the cubic B-spline basis functions associated with the knot vectors

$$\mathbf{s}_i = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$$

and

$$\mathbf{t}_i = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}]$$

respectively. The knot vectors \mathbf{s}_i and \mathbf{t}_i are extracted from the T-mesh neighborhood of P_i . Refer to [1, 5] for details on T-splines.

One of the most fundamental operations in T-splines is local refinement or local knot insertion, which means to insert one or more control points into a T-mesh without changing the shape of the T-spline surface. The algorithm of inserting new control point(s) can be described as follows:

Local knot insertion algorithm

- 1) Insert new control point(s) into the T-mesh.
- 2) If any basis function is missing a knot inferred from the current T-mesh, perform the necessary knot insertions into that basis function.
- 3) If any basis function has a knot that is not indicated in the current T-mesh, add an appropriate control point into the T-mesh.
- 4) Repeat steps 2) and 3) until there is no new operation. Now every basis function properly associates with a control point in the T-mesh.

The algorithm is based on the basis function refinement which in turn results in linear transformations on the control points. The main idea here is to maintain the validity of the T-mesh and to ensure that all the basis functions and the control points are properly associated.

2.2 Hierarchical B-splines

A hierarchical B-spline surface consists of a series of levels, each of which has a collection of B-spline patches. The level 0 has the lowest resolution describing the basic shape of the surface, and its children are called overlays describing some details of the surface. The overlays correspond to refined areas of the parent surface. An overlay at k level is created by designating a patch on its parent at level $k - 1$, executing a refinement step and manipulating the control points of the refined patch. Therefore an overlay at level k has twice the resolution of its parent at level $k - 1$. At the same level, the overlays do not cross each other. Otherwise, they are made into a larger overlay.

Note that to keep C^2 continuity between the overlay and its parent the manipulation of the control points should be restricted to the central control points and the peripheral control points should be kept static. For a bicubic B-spline surface, an overlay has a mesh of at least 7×7 and the control points on the three outer rings must remain unchanged. In the hierarchical B-spline scheme, a convenient way to represent the overlay is to use the reference-plus-offset form. That is, each control point $P^{(k)}$ of an overlay at level k is represented in offset form: $P^{(k)} = R^{(k)} + O^{(k)}$ where $R^{(k)}$ is derived from the parent surface at level $k - 1$ by a (mid-point) refinement and $O^{(k)}$ is the offset vector. Any change to the control points at level k is represented in the offset vectors. For those control points that remain unchanged, the offset $O^{(k)} = 0$. Offsets are responsible for the difference between the shape of the surface at two levels of representation. In this way, any changes to the surface at the lower level automatically propagate to the higher level and the editing movement of $O^{(k)}$ causes local change of the surface to a restricted region of the surface. Moreover, the reference-plus-offset form improves the economy of the representation because we only need to store the non-zero offsets. For example, an overlay of dimension 7×7 has only one non-zero offset.

In this paper, the concept of hierarchical B-splines is generalized in two aspects. First, the refinement in the hierarchical B-spline formulation is not necessarily mid-point refinement. Instead, the refinement can be achieved by allowing insertion of knots at general positions. As a result, a knot in the knot sequences of an overlay could be an arbitrary convex combination of two adjacent knots in the parent surface's knot sequences. Second, the B-spline surfaces could be rational. In this case, we apply the idea of original hierarchical B-spline refinement to the homogeneous representation of the rational B-splines. We still use the reference-plus-offset form for the overlays. Thus the offset vectors are 4D vectors. To maintain the continuity between

the overlay and its parent, we restrict all the offset vectors corresponding to the three outermost rings of control points of the overlay to be zero.

3 Converting a hierarchical B-spline to a T-spline

Given a hierarchical (rational) B-spline surface, one can flatten the hierarchical structure by creating the underlying (rational) B-spline surface, which usually contains a number of superfluous control points that serve no purpose other than to satisfy topological requirements of the tensor-product B-spline surface formulation. The transformation of a hierarchical B-spline surface into a T-spline surface is to combine all overlays into a single layer surface with only a few superfluous points and also to keep the resulting surface identical to the original one. Our approach to such a transformation is to use a recursive way to generate the T-spline surface: we begin with the level 0 B-spline surface and add all the first level overlay(s) to it to form a T-spline surface; then we add the second level overlay(s) into the formed T-spline to create a more complicated T-spline surface; and this process continues until there is no overlay left. Obviously, the key ingredient in this approach is the algorithm that adds an overlay into a T-spline surface. In the following we describe the algorithm.

Now suppose we have a T-spline surface $P(s, t)$ and an overlay $V(s, t)$. Assume $P(s, t)$ is defined by equation (1) and it is the result of combining all the overlays that have lower level than $V(s, t)$ and some of the overlays that have the same level as $V(s, t)$ into the level 0 B-spline surface. The overlay $V(s, t)$ is a bicubic rational B-spline surface, whose equation in homogeneous form is

$$V(s, t) = \sum_{i=l-1}^{r+1} \sum_{j=b-1}^{t+1} V_{ij} N_i(s) N_j(t) \quad (2)$$

where V_{ij} are control points in 4D space, $N_i(s)$ and $N_j(t)$ are cubic B-spline basis functions defined over knot sequences $\{s_{l-2}, s_{l-1}, s_l, \dots, s_r, s_{r+1}, s_{r+2}\}$ and $\{t_{b-2}, t_{b-1}, t_b, \dots, t_t, t_{t+1}, t_{t+2}\}$. The parameter range of the overlay is $[s_l, s_r] \times [t_b, t_t]$. From the formulation of the hierarchical B-spline, s_l, s_r, t_b and t_t are also the knots of $V(s, t)$'s parent surface and thus correspond to certain edges in the T-mesh of $P(s, t)$.

Based on the reference-plus-offset form, we only store the offset for the overlay. In fact, each point V_{ij} of the overlay is the sum of reference point R_{ij} and offset vector O_{ij} . All the reference points R_{ij} are specified from the parent surface. They form a surface

$$R(s, t) = \sum_{i=l-1}^{r+1} \sum_{j=b-1}^{t+1} R_{ij} N_i(s) N_j(t)$$

which is an exact re-representation of the parent surface or the T-spline surface $P(s, t)$ within domain $[s_l, s_r] \times [t_b, t_t]$. If we use

$$\text{all the offset vectors to define a 4D surface } O(s, t) = \sum_{i=l-1}^{r+1} \sum_{j=b-1}^{t+1} O_{ij} N_i(s) N_j(t), \text{ then } V(s, t) = R(s, t) +$$

$O(s, t)$. Also note that when $i < l+2, i > r-2, j < b+2$ or $j > t-2, O_{ij} = 0$. Therefore the overlay $V(s, t)$ can be written

$$V(s, t) = \sum_{i=1}^n P_i B_i(s, t) + \sum_{i=l+2}^{r-2} \sum_{j=b+2}^{t-2} O_{ij} N_i(s) N_j(t) \quad (3)$$

If we treat O_{ij} as a control point associated with knots (s_i, t_j) , equation (3) gives a PB-spline description of the overlay [1]. To make the PB-spline be a T-spline, we may need to perform the necessary knot insertions into the basis functions or add new control points into the mesh to make the mesh be a valid T-mesh [5]. This leads to the following algorithm for composing an overlay into a T-spline.

- 1) Insert points O_{ij} ($i = l+2, \dots, r-2; j = b+2, \dots, t-2$) with their respective basis functions into the T-mesh at the locations inferred by their associated knots.
- 2) If any basis function has a knot that is not indicated in the current T-mesh, add an appropriate point into the T-mesh.
- 3) If any basis function is missing a knot inferred from the current T-mesh, perform the necessary knot insertions into that basis function.
- 4) Repeat steps 2) and 3) until there is no new operation.
- 5) If any point has only one edge incident to it, extend the edge and find the first intersection point of the edge and the T-mesh in the pre-image space.
- 6) Insert the intersection point and then go to step 2).

Steps 5)-6) are used to avoid hanging edges.

Example. We illustrate the algorithm with an example. Fig. 1a shows a B-spline surface described by a 6×6 control mesh (in dash lines) and an overlay described by a refined 7×7 mesh (in solid lines). We first insert the nonzero offset point of the 7×7 mesh (that is the central point) into the 6×6 B-spline mesh, generating a mesh shown in Fig. 1b. However, the basis function corresponding to this nonzero offset point has knots that are not indicated in the current mesh. We have to add four control points around the nonzero offset point (see Fig. 1c). Now every basis function properly associates with a control point in the mesh. But we have four hanging edges in the mesh. To avoid hanging edges, we finally perform steps 5)-6), inserting another 4 points, to create the T-mesh as shown in Fig. 1d. It can be seen that the result T-mesh is quite compact.

4 Converting a T-spline to a hierarchical B-spline

Compared to the transformation from a hierarchical B-spline surface to a T-spline surface, the reverse transformation that converts a T-spline to a hierarchical (rational)

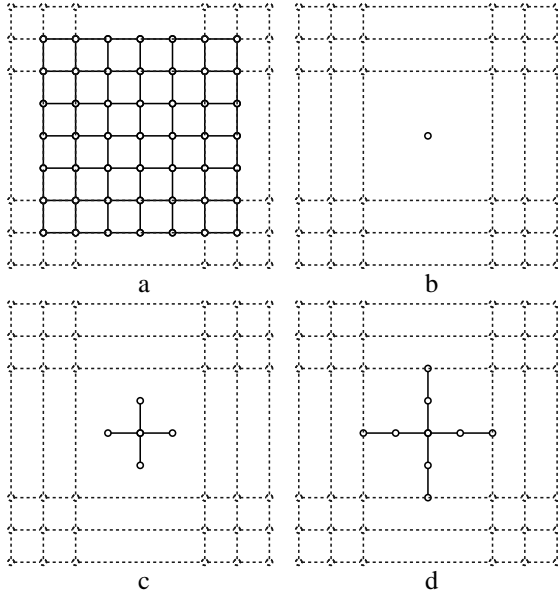


Figure 1. An example of converting a hierarchical B-spline to a T-spline

B-spline is somehow obscure since we intend to create a hierarchy from a single layer T-mesh. The basic task is to generate a series of B-spline surfaces. These B-splines are organized in a hierarchical manner. Topologically, except for the base B-spline, every B-spline is the refinement of another B-spline in the group. The process for this task involves two aspects: the topology and the geometry. The former identifies the topology of the base B-spline surface and the overlays at each level. The latter computes the coordinates of the control points for all the surfaces. Our algorithm roughly consists of the following steps:

- 0) Preprocess the given T-mesh.
- 1) Set $k = 0$, $currentMesh =$ the preprocessed T-mesh.
- 2) Extract a B-spline surface at level k from the $currentMesh$.
- 3) Determine the offset surfaces for level $lev = k + 1$.
- 4) for (the mesh of each offset surface at level lev) {
 - process the mesh;
 - if the processed mesh is not a B-spline mesh then
 - set $currentMesh =$ the processed mesh;
 - $k = lev$;
 - goto step 2)

The preprocess step is to use the T-spline local knot insertion algorithm to make the three outmost rings of the T-mesh not contain any T-junction points. Refer to Fig. 2 (left) for a given T-mesh. Fig. 2 (right) shows the mesh after the preprocess step. Step 1) just does some initialization. Now we present the details of the remaining steps below.

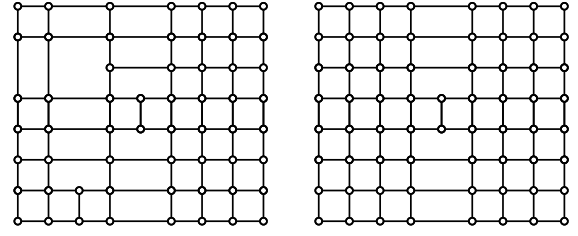


Figure 2. Preprocess a given T-mesh

4.1 Extract a B-spline

While a T-mesh permits T-junctions so that lines of the T-mesh need not traverse the entire control grid, a tensor-product B-spline mesh must be topologically a rectangular grid. Therefore to extract a B-spline mesh from a T-mesh, we eliminate all partial rows and all partial columns in the T-mesh. What remains after the removal of partial rows and columns is an $m \times n$ mesh, which can be used as the topological specification of our B-spline control mesh.

Once the topology of the B-spline mesh is specified, we have to compute the geometry of the B-spline mesh. While there might exist many solutions for the B-spline geometry, here we just present one solution. The formulae for the B-spline control points are derived based on polar form of B-splines [6]. The underlying requirement for the solution is to make the vector points on the three outer rings of the higher level offset determined in Section 4.2 be zero.

For each control point P in the extracted B-spline mesh (see Fig. 3a), we compute its polar label PL_P . Denote $PL_P = (s_{-1}, s_0, s_1) \times (t_{-1}, t_0, t_1)$. Point P corresponds to a control point, say Q , in the T-mesh. We also compute Q 's polar label PL_Q in the T-mesh. If $PL_P = PL_Q$, then we just copy Q 's coordinates to P 's. Otherwise, we create a temporary T-mesh by duplicating the existing T-mesh and use T-spline's local knot insertion algorithm to insert points in the temporary T-mesh, forming a 5×5 grid centered around Q . This 5×5 grid is part of the finest control mesh of the underlying B-spline surface. We denote the points by Q_{ij} , ($i, j = -2, \dots, 2$) with Q_{00} corresponding to Q . Refer to Fig. 3b for labels. Then P can be computed by

$$P = \sum_{i=-1}^1 \sum_{j=-1}^1 c_i d_j Q_{ij}$$

where

$$\begin{aligned} c_{-1} &= \frac{(a_1 - s_{-1})(a_1 - s_1)}{(a_1 - a_{-2})(a_1 - a_{-1})} \\ c_1 &= \frac{(a_{-1} - s_{-1})(a_{-1} - s_1)}{(a_2 - a_{-1})(a_1 - a_{-1})} \\ c_0 &= 1 - c_{-1} - c_1 \end{aligned}$$

and d_i are similar to c_i where a_j should be replaced by b_j and s_j should be replaced by t_j .

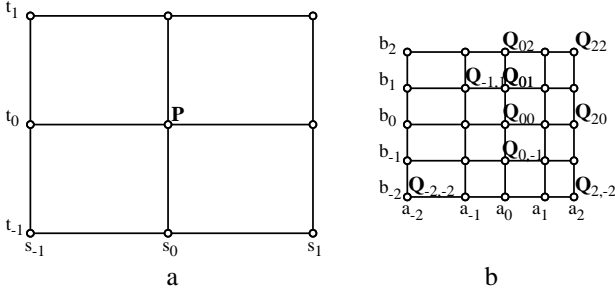


Figure 3. Compute B-spline control point P from Q_{ij}

4.2 Determine higher level offsets

When we have extracted a B-spline mesh from a T-mesh at level k , both the B-spline mesh and the T-mesh define two surfaces which are not necessarily the same. In this case, we need higher level (offset) surfaces to compensate the difference.

We first locate the region for the higher level offset surfaces. For each point P in the pre-image of the T-mesh at level k in (s, t) parameter space, we define four numbers s_min, t_min, s_max and t_max such that s_min is the s coordinate of a new point which is obtained by moving point P left by two bays, s_max is the s coordinate of a point which is obtained by moving P right by two bays, and t_min, t_max are similarly defined. For each partial row or column in the T-mesh, we define its box-2 $O(s_l, t_b, s_r, t_u)$ to be a rectangle specified by the lower-left corner (s_l, t_b) and the upper-right corner (s_r, t_u) where s_l, t_d are the minima of s_min and t_min of all vertices lying on the partial row or column, and s_r, t_u are the maxima of s_max and t_max of all vertices lying on the partial row or column. Now we compute box-2 for all partial rows and columns. If any two box-2's overlap, we merge them to form a large box bounding them. We also call the large box a box-2. This large box is used to replace the original two small boxes. After this, each box-2 stands for a region of an overlay.

Once the region of an offset surface in the next layer is identified, we have to determine topology and geometry of the offset surface. We extend the box-2 by adding one more ring. The pre-image corresponds one-to-one with the T-mesh. We take the portion of the T-mesh at level k , which corresponds with the extended box-2. It forms a temporary T-mesh at level $k + 1$ for the surface within the identified region. By the construction of box-2, it can be verified that the level $k + 1$ T-mesh has such characteristic that the three outermost rings of the mesh do not contain T-junctions.

The 4D surface defined by the T-mesh at level k within the region is described by

$$P^{(k)}(s, t) = \sum_{i=1}^n P_i^{(k)} B_i(s, t) \quad (4)$$

where $P^{(k)}$ are the control points in the T-mesh at level k

or in the temporary T-mesh at level $k + 1$. In order to get the offset surface, we restrict the extracted B-spline surface (from the T-mesh at level k) within the region and write its equation to be

$$R^{(k)}(s, t) = \sum_{i=l-1}^{r+1} \sum_{j=b-1}^{t+1} R_{ij}^{(k)} N_i(s) N_j(t) \quad (5)$$

Then the offset surface is $P^{(k)}(s, t) - R^{(k)}(s, t)$. This gives the following algorithm for determining the topology and geometry of the offset surface.

- 1) Insert points $-R_{ij}^{(k)}$ ($i = l - 1, \dots, r + 1; j = b - 1, \dots, t + 1$) with their respective basis functions into the temporary T-mesh at the locations inferred by $R^{(k)}_{ij}$'s associated knots.
- 2) If any basis function is missing a knot inferred from the current T-mesh, perform the necessary knot insertions into that basis function.
- 3) If any basis function has a knot that is not indicated in the current T-mesh, add an appropriate point into the T-mesh.
- 4) Repeat steps 2) and 3) until there is no new operation.
- 5) The output mesh is the required T-mesh at level $k + 1$ which defines the offset surface.

4.3 Process a layer

Now we have obtained the T-mesh at level $k + 1$ for the offset surface. If it happens to be a B-spline mesh, it is a leaf node in the hierarchical B-spline representation and we do not need to do further process. Otherwise, some further process is required. We can either extend all partial rows and columns all the way to create a B-spline or further extract higher level of layers.

We take a strategy to balance the depth of the hierarchy and size of each offset. For each partial row or column in the T-mesh at level $k + 1$, we calculate a number, called *deficiency*, which measures how many knots in the underlying B-spline this partial row or column needs to go through to reach the border. If the deficiency is large, the row or column needs to take a long way to reach the border, which might mean that it depicts details and thus would be better to be put in the overlay of the current layer. For this reason, we compute the average of the deficiencies of all partial rows and columns. If any partial row or column has a deficiency less than the average deficiency plus 3, we extend it all the way to make it a complete row or column.

If there is no T-junction after the extension, then a B-spline is formed and can be treated as a leaf node attached to the parent surface. If there still exist T-junctions, then pass the mesh for further decomposition in the next level.

