A Fast T-spline Fitting Method based on Feature Extraction for Large-size Z-map Data

Yazui Liu[1], Gang Zhao[2], Xiaoxiao Du[3], Wei Wang[4] and Aizeng Wang[5]

[1]Beihang University, liuyazui@buaa.edu.cn
[2]Beihang University, zhaog@buaa.edu.cn
[3]Beihang University, duxiaoxiao@buaa.edu.cn
[4]Beihang University, jrrt@buaa.edu.cn
[5]Beihang University, azwang@buaa.edu.cn

Corresponding author: Gang Zhao, zhaog@buaa.edu.cn

Abstract. Data fitting is a fundamental tool for constructing a smooth representation from 3D data (Z-map data) in computer-aided design, computer graphics, and reverse engineering. T-spline has been widely adopted for complex data fitting with the advantages of fewer control points, local refinement, and watertight representation. However, the T-spline fitting for Z-map data is inefficient by using a traditional two-phase iterative method, which requests updating T-mesh and recomputing all control points in each iteration. Hence, the traditional T-spline fitting method is time-consuming for large-size Z-map data reconstruction that is widely used in high resolution image processing, geographic information system, and scientific data visualization. In this paper, a fast T-spline fitting method is proposed based on feature extraction for large-size Z-map data. Feature extraction is introduced to construct the ultimate T-spline control grid based on T-spline local refinement without iteration, and an efficient progressive iterative fitting method is employed for T-spline control points evaluation. Computing costs can be reduced obviously since the proposed method is a single-phase iterative method. The proposed method is demonstrated using two types of large-size Z-map data.

Keywords: T-spline; data fitting; feature extraction; Z-map data.
DOI: https://doi.org/10.14733/cadaps.2023.261-274

1 INTRODUCTION

In recent years, the surface of the real objects can be easily measured or scanned with the development of data capture devices, and the measurement data become larger and larger for high precision and excellent quality. The measurement data represented using a two-dimensional regular grid with height values is called Z-map data that is convenient for storage. Z-map data is widely used for high resolution image processing, geographic information system, and scientific data visualization. However, Z-map data is a type of discrete representation, which can be applied only to some types of objects, and is rarely used for the downstream industry applications directly such as modeling, 3D printing, toolpath generation, and reverse engineering. Hence, Z-map data need to be further processed for the end-use of downstream applications. Typically, data fitting technology is widely used to convert Z-map data into the parametric surface for the purpose of CAD/CAE/CAM usage, which has the capability of size reduction, smooth functionality, high-performance rendering, and noise insensitive.

Non-uniform rational B-spline (NURBS) has been an industry standard in CAD/CAE/CAM[14], which is an excellent tool for surface fitting. Nevertheless, many superfluous control points will be introduced to only satisfy the topological structure of NURBS, especially for fitting large-size Z-map data, which can decrease the computing efficiency. The multi-level NURBS fitting methods are investigated to fit surface from coarse to fine levels [4][10], which also introducing redundant control points. To address the shortcoming of NURBS inherently, T-spline [15][16] and its variations [1][5-7] are proposed with flexible topological structure and outstanding mathematical properties. T-spline allows control points to terminate at arbitrary row or column, forming T-junctions. Hence it is able to significantly avoid the superfluous control points, especially for large-size Z-map data fitting.

T-spline fitting is an important application and has been researched widely in the last decade. T-spline was first introduced for Z-map data fitting by use of local refinement iteratively, and a discrete fairness function was given to obtain a smooth surface [22]. T-spline level set was used to capture the topology and outline for the large set of unorganized point sample data reconstruction [19]. An algorithm of constructing periodic T-spline surface was presented to approximate tubular triangular meshes [21], and closed T-spline was introduced to reconstruct a surface from medical image data [17]. T-spline surface skinning was investigated to fit rows of data points [20], and local T-spline surface skinning methods were studied to meet additional constraints [11][12]. A curvature-guided T-spline fitting method was proposed to effectively capture model features based on faithful re-parametrization and initial knot re-placement [18]. The T-spline fitting methods mentioned above are based on the least-square fitting algorithm, which will introduce a large amount of computing cost for T-spline control points re-computing in each iteration, and the computing is invalid in some cases.

The fast T-spline fitting methods were researched to improve computing efficiency, especially for large-size data sets. A progressive T-spline fitting algorithm was proposed for large image fitting, and its iteration speed is steady and insensitive to the increasing number of unknown T-spline control points [8]. A novel split-connect-fit T-spline fitting algorithm was presented for large-size point clouds, which divides the point clouds into a set of B-spline patches and connect them into a single T-spline surface [2]. A fast T-spline method was studied based on segmentation, which divides the mesh into inactive and active parts, and only the control points in the active part need to be recalculated in the upcoming process [9].

How to find a proper T-mesh that describes the topology of the T-spline surface and how to achieve optimal control points that decide the geometry of the T-spline surface are two critical points for T-spline fitting. According to the described methods above, T-spline fitting employed a two-phase iterative method, which means the T-mesh and the control points need to be both updated in each iteration. Hence, the dual processing will be time-consuming for fitting large-size data sets. In the paper, a single-phase iterative method is proposed for T-spline fitting based on feature extraction, which can improve the computing efficiency significantly for large-size Z-map data. The T-mesh is initialized once using feature information extracted from Z-map data without updating, and only control points need to be re-computed iteratively.

The rest of the paper is organized as follows. An overview of T-spline and T-spline fitting methods are presented in Section 2. In section 3, a fast T-spline fitting method based on feature extraction is proposed, and some optimized algorithm is discussed to further improve computing efficiency. Section 4 provides T-spline fitting results for two types of large-size Z-map data, and the comparison with existing methods is discussed. The conclusion and future work are given in Section 5.

2 RELATED WORK

2.1 T-spline

A T-spline surface is defined by a control grid called T-mesh, which allows control point to terminate in a partial row or column named as T-junction. T-spline enables to add control points locally without propagating an entire row or column, and the number of control points is reduced significantly without altering the corresponding NURBS surface under the given tolerance. T-spline is a point-based spline that usually defined using the following equation:

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

where $P_i=(x_i,y_i,z_i,w_i)$ are the control points in homogeneous coordinate system, and $B_i(s,t)$ are the T-spline blending functions given by the following equation:

$$B_i(s,t)=N_i(s)\bullet N_i(t) \tag{2}$$

$N_i(s)$ and $N_i(t)$ are B-spline basis functions calculated iteratively as follows:

$$\begin{cases} N_{i,0}(t)=\begin{cases}1, t\in[t_i,t_{i+1}]\\ 0, otherwise\end{cases} \\ N_{i,k}(t)=\dfrac{t-t_i}{t_{i+k}-t_i}N_{i,k-1}(t)+\dfrac{t_{i+k+1}-t}{t_{i+k+1}-t_{i+1}}N_{i+1,k-1}(t)\end{cases} \tag{3}$$

In the paper, *k* is set to 3 in equation (3) for the calculation of bi-cubic T-spline surface, and the B-spline basis function $N_i(s)$ and $N_i(t)$ are defined using five knots respectively:

$$\begin{cases} N_i(s)=N[s_{i0},s_{i1},s_{i2},s_{i3},s_{i4}](s)\\ N_i(t)=N[t_{i0},t_{i1},t_{i2},t_{i3},t_{i4}](t)\end{cases} \tag{4}$$

(s_{i2},t_{i2}) is the coordinate of control point P_i, and its knot vectors in each direction can be extracted using ray intersection method [15].

2.2 T-spline Fitting

T-spline fitting method was first introduced in [22]. The T-spline fitting problem can roughly be stated: considering a set of data points $Q=Q_1,Q_2,\ldots,Q_m$, we look for a T-spline surface $S(s,t)$ to satisfy the equation below for each fitting point Q_i:

$$\left\|Q_i-S(s_i,t_i)\right\|<\varepsilon \tag{5}$$

where ε is an error tolerance specified by user, and (s_i,t_i) is the parameter for Q_i on T-spline surface.

The equation (5) can be rewritten as:

$$A^T AP=A^T Q \tag{6}$$

and

$$\begin{cases} A=\begin{bmatrix} B_1(s_1,t_1) & B_2(s_1,t_1) & \cdots & B_n(s_1,t_1) \\ B_1(s_2,t_2) & B_2(s_2,t_2) & \cdots & B_n(s_2,t_2) \\ \cdots & \cdots & \cdots & \cdots \\ B_1(s_m,t_m) & B_2(s_m,t_m) & \cdots & B_n(s_m,t_m) \end{bmatrix} \\ P=\begin{bmatrix} P_1 \\ P_2 \\ \cdots \\ P_n \end{bmatrix} Q=\begin{bmatrix} Q_1 \\ Q_2 \\ \cdots \\ Q_n \end{bmatrix} \end{cases} \tag{7}$$

The control points of T-spline fitting surface can be solved by $P=\left(A^T AP\right)^{-1}Q$ that is a least-square system, which could be invalid and time-consuming for fitting large-size data sets.

Progressive iterative fitting method is proposed for T-spline to overcome the difficulties above [8]. Due to the local support of T-spline blending functions, the speed of progressive iterative fitting is steady and insensitive to the increasing number of T-spline control points, which is able to efficiently fit large-size data sets. The (k+1)st control point P_i^{k+1} is updated by *adjusted vector* (Δ_i^k) and P_i^k as follows:

$$P_i^{k+1}=P_i^k+\Delta_i^k \tag{8}$$

wherein

$$\begin{cases} \Delta_i^k=\dfrac{\sum_{j=1}^{m} B_i\left(s_j,t_j\right)\delta_j^k}{\sum_{j=1}^{m} B_i\left(s_j,t_j\right)} \\ \delta_j^k=\overrightarrow{R^k\left(s_j,t_j\right)Q_j} \end{cases} \tag{9}$$

Q_j is the fitting point, $\left(s_j,t_j\right)$ is its parameter on the T-spline surface, $R^k\left(s_j,t_j\right)$ is the corresponding point on the surface for the fitting point, δ_j^k is the *difference vector* calculated between each fitting point and its corresponding point on the T-spline surface, and the *difference vector* is distributed for the blending function of the control points which has non-negative contribute for the fitting point. The details are discussed in next section.

Once current T-spline is computed, tolerance checking is executed to decide whether the result is satisfactory. T-mesh should be updated over the regions whose error is greater than the tolerance based on T-spline local refinement. Assuming the region is bounded by a box defined by whose lower-left corner $\left(s_{min},t_{min}\right)$ and upper-right corner $\left(s_{max},t_{max}\right)$. If $s_{max}-s_{min}>t_{max}-t_{min}$, then the region is split at $s=\left(s_{max}+s_{min}\right)/2$. If $s_{max}-s_{min}<t_{max}-t_{min}$, then region is split at $t=\left(t_{max}+t_{min}\right)/2$. If $s_{max}-s_{min}==t_{max}-t_{min}$, then either direction could be selected for region splitting.

3 THE PROPOSED ALGORITHM

In the section, a single-phase iterative T-spline fitting method is presented based on feature extraction. Comparing with the traditional two-phase iterative T-spline fitting method, the final T-mesh can be constructed once without updating, which is more efficient for large-size Z-map data fitting. An efficient progressive iterative T-spline fitting method is also presented.

3.1 T-mesh Construction

3.1.1 Parametrization of Z-map data

In the paper, we focus on Z-map data that is a discrete representation based on rectangular grid. Each fitting point on the Z-map grid is defined by $\{(i,j),Q(i,j)\}$, (i,j) is specified as the two-dimensional coordinate value on Z-map grid, and $Q(i,j)$ is the height value with respect to (i,j). Parametrization is a process of finding a mapping that assign each point on Z-map grid a pair of parameter values (s_i,t_j). Many parametrization algorithms have been investigated [3]. Z-map data is topologically equivalent to a rectangle, and the distance between each fitting point is equal, hence uniform parametrization method is employed in the paper.

We denote $Q=\{Q(i,j);i=1,\ldots,m,j=1,\ldots,n\}$ as the Z-map data, *m* and *n* are the size of Z-map data in each direction, then the parameter (s_i,t_j) for each fitting point on Z-map grid can be obtained intuitively as below:

$$\begin{cases} s_i=(i-1)/(m-1), i\subseteq[1,m] \\ t_j=(j-1)/(n-1), j\subseteq[1,n] \end{cases} \tag{10}$$

3.1.2 Initial of T-mesh

The bicubic B-spline patch with a 4×4 control grid is used as the initial T-mesh generally [2][9] [18][22], that will spend much more iterative times to find a proper T-mesh, especially for large-size Z-map data. Hence, a bi-cubic B-spline patch with $max\left(\left\lfloor 20*log_2^{\frac{m*n}{\theta}}\right\rfloor,4\right)\times max\left(\left\lfloor 20*log_2^{\frac{m*n}{\theta}}\right\rfloor,4\right)$ control points is adopted as the initial T-mesh in the paper to reduce the iterative times, $\lfloor x\rfloor$ denotes the integer number closest to x, and θ is specified by user to adjust the density of T-mesh. A dense T-mesh is initialized when smaller θ is used, which will spend less time with introducing many superfluous control points, and vice verse. The initial T-mesh consists of a set of faces, and T-spline local refinement is performed in the next section based on face split.

3.1.3 T-mesh construction based on feature extraction

The uniform T-mesh grid is initialized that can be used to fit flatness parts of the Z-map data. To improve the fitting quality, additional control points need to be inserted over the steep parts, which have greater fitting error normally. Edge detection is one of the key fundamental tools for image processing, which can capture sharp changes in the image. In the paper, edge detection technology is introduced for Z-map data feature extraction due to its similar topology to image data. A huge number of methods is investigated for edge detection, which uses first-order or second-order derivative typically. Among the edge detection methods proposed so far, the Canny edge detector is widely used due to three criteria of good detection, good localization, and single response to an edge. A typical implementation of the Canny edge detection is presented below, and is easily re-implemented for Z-map data:

1) Apply Gaussian filter to smooth the data in order to reduce data noise.
2) Determine gradient magnitude and gradient direction for each data point.
3) Apply non-maximum suppression to get rid of spurious response to edge detection.
4) Apply double threshold to determine potential edges.
5) Remove the weak edges by hysteresis thresholding.

The detected edges consist of a set of discrete points that are chosen as the candidate inserted points to perform T-spline local refinement for the initial T-mesh. The simple rule is used to assign the candidate inserted points to their corresponding face of the initial T-mesh. We denote Q_k is one of the candidate's inserted points, (s_k, t_k) is its parameter in the initial T-mesh, and $(s_i, s_{i+1}) \otimes (t_j, t_{j+1})$ is employed to describe the arbitrary face of the initial T-mesh. The candidate inserted point is assigned to the face when the equation is satisfied bellow:

$$\begin{cases} s_i \le s_k \le s_{i+1} \\ t_j \le t_k \le t_{j+1} \end{cases} \tag{11}$$

A multi-split T-spline local refinement algorithm is introduced for all faces that have candidate inserted points as shown in *Algorithm 1*.

```
Algorithm 1 A multi-split T-spline local refinement algorithm.
Input: $F_{old}$
Output: $F_{new}$
Step 1. Denote $Q_k, k \subseteq (1,n)$ are the candidate inserted points belong to $F_{old}$, sort $Q_k$ in descending order by fitting error. Initialize $F_{new}$ by $F_{old}$.
Step 2. Travel all candidate inserted points $Q_k$ iteratively.
  Step 2.1. Find the face $F_k$ that contains $Q_k$, split $F_k$ into $F_{k1}$ and $F_{k2}$ based on the length of the face. (see section 2)
  Step 2.2. Remove $F_k$ from $F_{new}$, and add $F_{k1}$ and $F_{k2}$ into $F_{new}$.
```

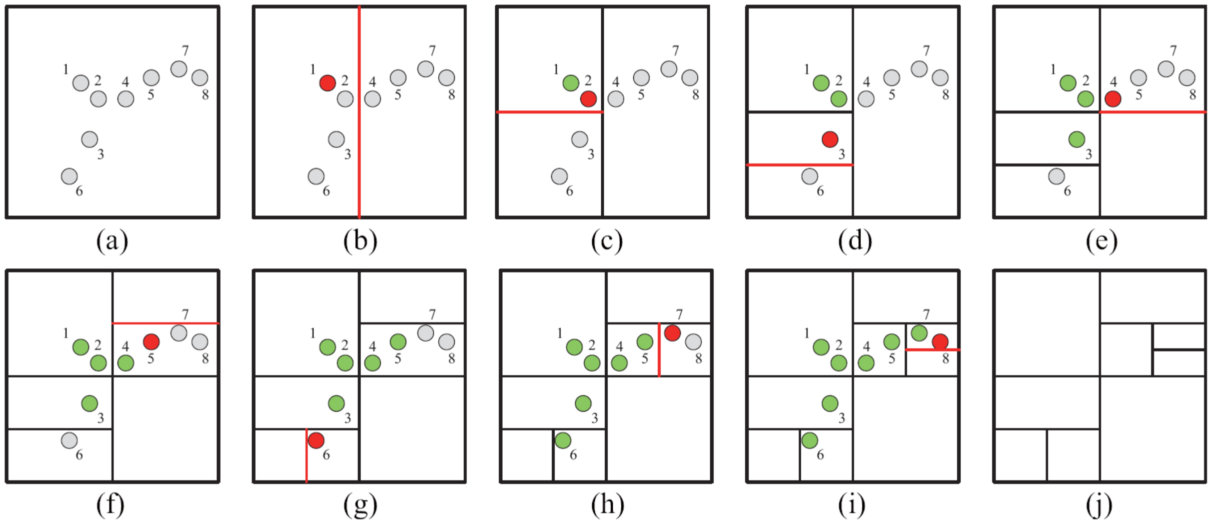

Figure 1. An example of the multi-split T-spline local refinement algorithm: (a) presents one face with its candidate inserted points sorted by the fitting error, (b) - (i) presents the results of each step of the algorithm, the unprocessed points are marked in gray, the current selected point is marked in red, the processed points are marked in green, the red line is the split edge for the current face, (j) gives the final face after executing the algorithm.

As shown in Figure 1. Figure 1(a) shows one of the faces with its candidate inserted points. Figure 1(b) - (i) present the procedure of the multi-split T-spline local refinement algorithm. Figure 1(j) displays the final face after executing the multi-split T-spline local refinement algorithm.

Three parameters *low threshold, high threshold* and *sigma* are specified by the user to control the results of Canny edge detection, and the three parameters are critical for T-mesh construction in the same manner. If the parameters are greater, some detail features are extracted, and much more control points are introduced into T-mesh to obtain high quality result, and vice verse. The three parameters should be selected carefully based on the structure of Z-map data and its application.

The final T-mesh is constructed once based on feature extraction to describe the topology of Z-map data without iteration, which will save a lot of time comparing with the existing methods.

3.2 Control Points Calculation

In this section, the concepts of control point influence domain and face-influenced points are introduced to improve the computing efficiency, and an efficient progressive iterative fitting method is presented based on T-spline local support.

3.2.1 The influence domain of T-spline control point

For each of T-spline control points P_i, the ray intersection method is adopted to determine its two local knot vectors as below:

$$\begin{cases} S_i = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}] \\ T_i = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}] \end{cases} \tag{12}$$

Each set of knot vectors defines a T-spline blending function that has a non-negative value over a rectangular area $[s_{i0}, s_{i4}] \otimes [t_{i0}, t_{i4}]$, which is called the influence domain of the control point.

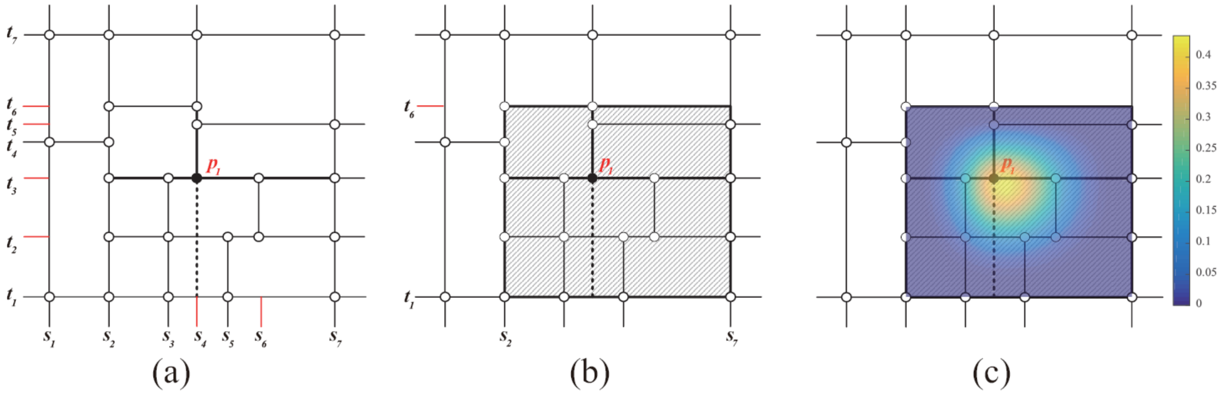

Figure 2: (a) gives an example of T-mesh with its parametric space, (b) displays the influence domain of control point *P*$_1$, (c) presents the blending function of control point *P*$_1$.

As shown in Figure 2. Figure 2(a) gives a T-mesh with its parametric space, the local knot vectors of control point P_1 are obtained intuitively as follows: $[s_2, s_3, s_4, s_6, s_7]$, $[t_1, t_2, t_3, t_5, t_6]$. Figure 2(b) shows the influence domain of T-spline control point P_1 defined by $[s_2, s_7] \otimes [t_1, t_6]$. Figure 2(c) shows the T-spline blending function of P_1. It is intuitive to show that the T-spline blending function of the control point only have non-negative value over its influence domain, and have no contribution for other parts.

3.2.2 Face-influenced points

T-spline consists of a set of faces that can be defined by a rectangular parametric area, and an overlap operation can be executed between the rectangular parametric area and the influence domain of T-spline control points. The control point has a non-negative contribution over the face if the overlap is existent, and we call it the face-influenced point. All face-influenced points are found when the overlap operation is accomplished for all T-spline control points. Face-influenced points provide the capability of local calculation for T-spline surface, which can be adopted for progressive iterative T-spline fitting efficiently. In addition to this, face-influenced points are independent for each face, and parallel computing can be employed to further improve efficiency.

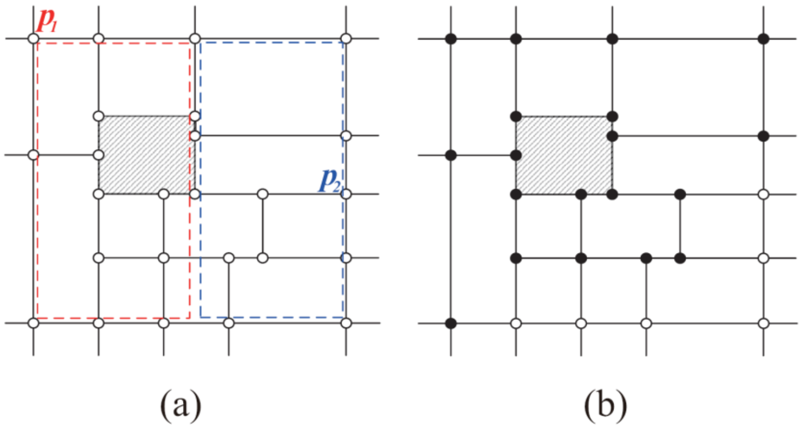

Figure 3: (a) The relationship between the face and the influence domain of control points, red rectangle is the influence domain of P_1 that has overlap over the shaded face, and blue rectangle is the influence domain of P_2 that doesn't have overlap over the shaded face, (b) gives all face-influenced points of the shaded face marked by black solid circle.

As shown in Figure 3(a), the influence domain of T-spline control point P_1 has overlap over the shaded face, and the influence domain of P_2 doesn't have overlap over the shaded face. Figure 3 (b) shows all face-influenced points of the shaded face marked by black solid circle.

3.2.3 An efficient progressive iterative T-spline fitting method

In this section, an efficient progressive iterative T-spline fitting method is presented based on the extraordinary properties of T-spline to satisfy the need of large-size Z-map data fitting. The flow chart of the proposed method is shown in Figure 4.

All fitting points are processed iteratively, for one of the fitting points Q_j, find the face *f* that contains the fitting points using equation (11). The face-influenced points of *f* are employed to calculate its corresponding point on the T-spline surface, and the *difference vector* is obtained and assigned for all face-influenced points of *f*. The *adjusted vectors* for all control points are acquired when the iteration is finished, and the new control points for the next fitting stage are computed straightway.

The Root Mean Square error is employed to describe the fitting error and is defined as follows:

$$\varepsilon^k = \sqrt{\frac{\sum_j^M \left\| R^k\left(s_j,t_j\right) - Q_j \right\|}{M}} \tag{13}$$

where $R^k\left(s_j,t_j\right)$ is the corresponding point on the surface for the fitting point, Q_j is the fitting point, *M* is the number of all fitting points. The progressive fitting iteration is terminated when either $\left|\frac{\varepsilon^{k-1}}{\varepsilon^k} - 1\right| < \eta$ or the iteration number exceeds a specified value N_{it}. We take $\eta = 0.005$, and $N_{it} = 20$ for our experiments.

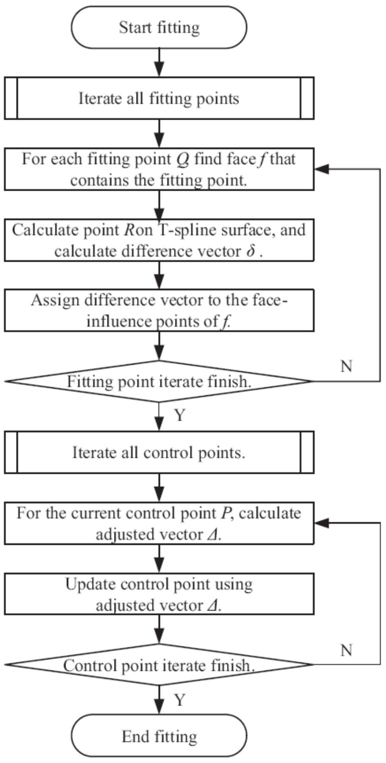
Start fitting
Iterate all fitting points
For each fitting point Q find face f that contains the fitting point.
Calculate point R on T-spline surface, and calculate difference vector δ.
Assign difference vector to the face-influence points of f.
Fitting point iterate finish.
N
Y
Iterate all control points.
For the current control point P, calculate adjusted vector Δ.
Update control point using adjusted vector Δ.
Control point iterate finish.
N
Y
End fitting

Figure 4: The flow chart of the proposed efficient progressive iterative T-spline fitting method.

4 EXPERIMENTS AND DISCUSSION

Our proposed algorithm is implemented in the C++ programming language with OpenMP parallelization and runs it on the desk PC with Intel Core i7 CPU of 3.4GHz, which consists of parametrization of Z-map data, ultimate T-mesh construction based on Canny edge detection, and an efficient progressive iterative fitting method based on T-spline local support. Two kinds of large-size Z-map data are considered that include high resolution image data and digital elevation model data. The speed and accuracy of the proposed algorithm are tested comparing with the methods proposed by Lin [8] and Feng [2].

4.1 High Resolution Image Data

The image is a typical Z-map data with RGB channels, and the resolution becomes higher and higher with the development of the capture device. The fitting results could help various image processing algorithms such as compressing, multi-resolution display, and geometric transformations. The fitting time (contains the time of feature extraction) and the number of T-spline control points are considered for the efficiency demonstration of our proposed algorithm. In addition to this, Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM) are employed to evaluate the quality of the fitting image. The fitting results and corresponding T-mesh are shown in Figure 5, and Table 1 shows the statistic information (the statistic information is obtained by the average of ten times fitting experiments).

Two images are used for comparison with reference [8] and [2], the comparison results are shown in Table 2. For the two cases, our algorithm is faster than the other two methods because of the single-phase iterative solution. Lin2013 obtains the best image quality, whereas it has more control points and lower computing efficiency. Feng2017 produces similar results with fewer control points than our algorithm due to the segmentation processing, which needs to consider the continuities between different patches.

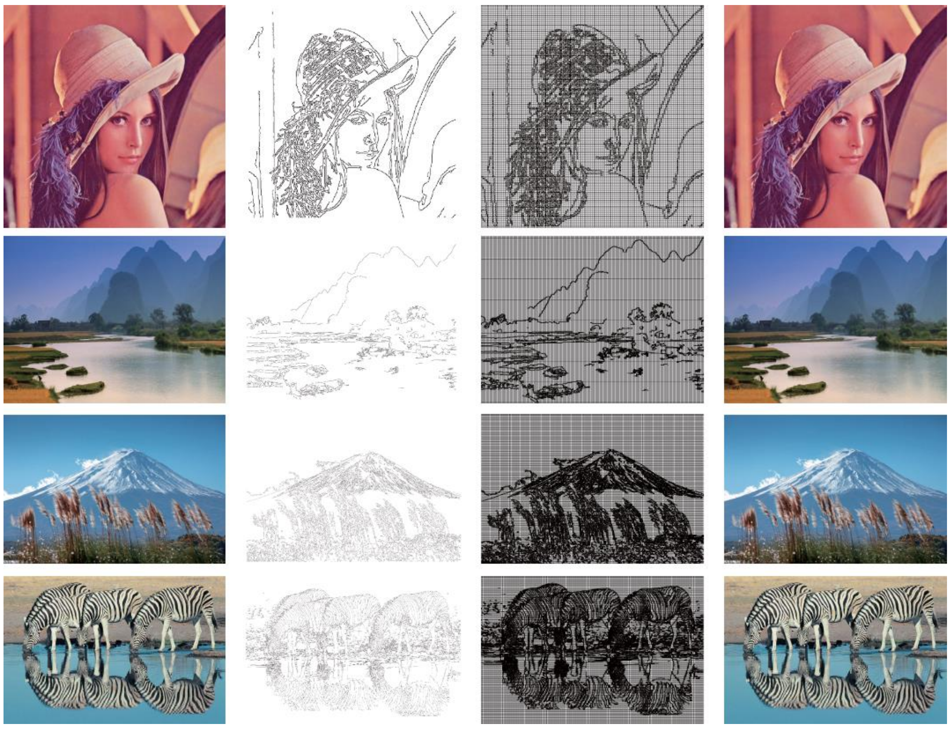

Figure 5: Fitting high resolution image data. From left to right: raw image, feature extraction results, T-mesh, fitting image.

	Resolution	*Control points*	*Times(s)*	*PSNR*	*SSIM*
Figure 5.1 Lenna	*512*512*	*40891*	*0.97*	*35.0187*	*0.9048*
Figure 5.2 Landscape	*1600*1200*	*109884*	*4.436*	*37.2658*	*0.8894*
Figure 5.3 Fuji	3000*2000	468726	24.599	35.8348	0.9162
Figure 5.4 Zebra	3660*2440	792673	40.768	32.4634	0.8713

Table 1: Image data statistic information for the proposed algorithm.

	Control points	*Times(s)*	*PSNR*
*Lenna(512*512)*			
Our algorithm	*40891*	*0.97*	*35.0187*

Lin2013	*41394*	*198.6*	*44.8396*
Feng2017	30533	1.18	32.5276
*Landscape(1600*1200)*			
Our algorithm	109884	4.436	37.2658
Lin2013	180211	2760	46.5826
Feng2017	43038	6.07	37.2496

Table 2: Comparison between our algorithm, Lin2013[8] and Feng2017[2] for image fitting.

4.2 Digital Elevation Model Data

Digital elevation model (DEM) data are often used in geographic information that can be represented as a raster-based model or as a vector-based model. A grid of squares is adopted for the geometrical representation, and the height number is used as the Z direction value for DEM data. There are three critical statistics for this comparison: fitting time, RMSE (Root Mean Squared Error), and the number of control points. The DEM data is available for download from OpenTopography [13]. The results are shown in Figure 6, and the statistic information are given in Table 3. The DEM results require more control points than image data due to it has more discontinuities areas, which need denser initial T-mesh and computing cost.

The fitting results demonstrate the efficiency and accuracy, for a Z-map data with about 3000*2000 resolution, the fitting time is less than 60 seconds with good quality in the paper. The quality of the fitting result and the fitting time are positively associated with the number of initial T-mesh and the feature extraction parameters separately. As shown in Table 4. In the paper, the initial T-mesh and the feature extraction parameters are specified by the user manually. How to select the optimal parameters automatically is one of the key points in our future work.

	Resolution	*Control points*	*Times(s)*	*RSME(m)*
Figure 6.1	*512*512*	*106229*	*1.691*	*10.628*
Figure 6.2	*1699*1572*	*232933*	*4.748*	*8.609*
Figure 6.3	1853*1701	410743	16.613	17.527
Figure 6.4	2048*2048	395260	12.318	8.858

Table 3: DEM data statistic information for the proposed algorithm.

resolution	*information*	*ls=0 hs=0.15 s=2.0[1]*	*ls=0.05 hs=0.2 s=1*	*ls=0 hs=0.15 s=1.5*	*ls=0.05 hs=0.2 s=0.8*	*ls=0 hs=0.15 s=1.0*
*64*64*	*cp[2]*	*28810*	*30915*	*31710*	*33285*	*40891*
	time[3]	*0.8274*	*0.8516*	*0.8597*	*0.885*	*0.97*
	PSNR	33.7318	34.2833	34.3669	34.7013	35.0187
	SSIM	0.8812	0.8891	0.8896	0.8950	0.9048
*80*80*	*cp*	31601	33830	34687	36140	43891
	time	0.8628	0.8698	0.8940	0.9247	1.0109
	PSNR	34.5125	35.0702	35.3275	35.6392	35.8904
	SSIM	0.8913	0.8985	0.9015	0.9062	0.9136
*100*100*	*cp*	41568	37467	38465	39798	47613
	time	0.8707	0.8996	0.9030	0.9327	1.0347
	PSNR	34.6159	35.1411	35.5108	35.8818	36.0107
	SSIM	0.8965	0.9031	0.9074	0.9127	0.9138
*128*128*	*cp*	41568	43836	44602	46238	54015

	time	0.8729	0.9026	0.9445	0.9458	1.0641
	PSNR	35.0129	35.4932	35.8596	36.1618	36.3067
	SSIM	0.9056	0.9114	0.9150	0.9189	0.9247
*150*150*	*cp*	48916	50964	52007	53445	61525
	time	0.9359	0.9682	1.0099	1.0185	1.144
	PSNR	35.1423	36.7247	37.0222	37.3998	37.5153
	SSIM	0.9181	0.9237	0.9261	0.9300	0.9345

Table 4: Lenna comparison results with different initial T-mesh and feature extraction parameters. [1] The ls denotes low threshold, hs denotes high threshold, s denotes the sigma for feature extraction.[2] The number of the control points of the T-mesh.[3] The fitting time (seconds) is obtained by the average of ten times fitting experiments.

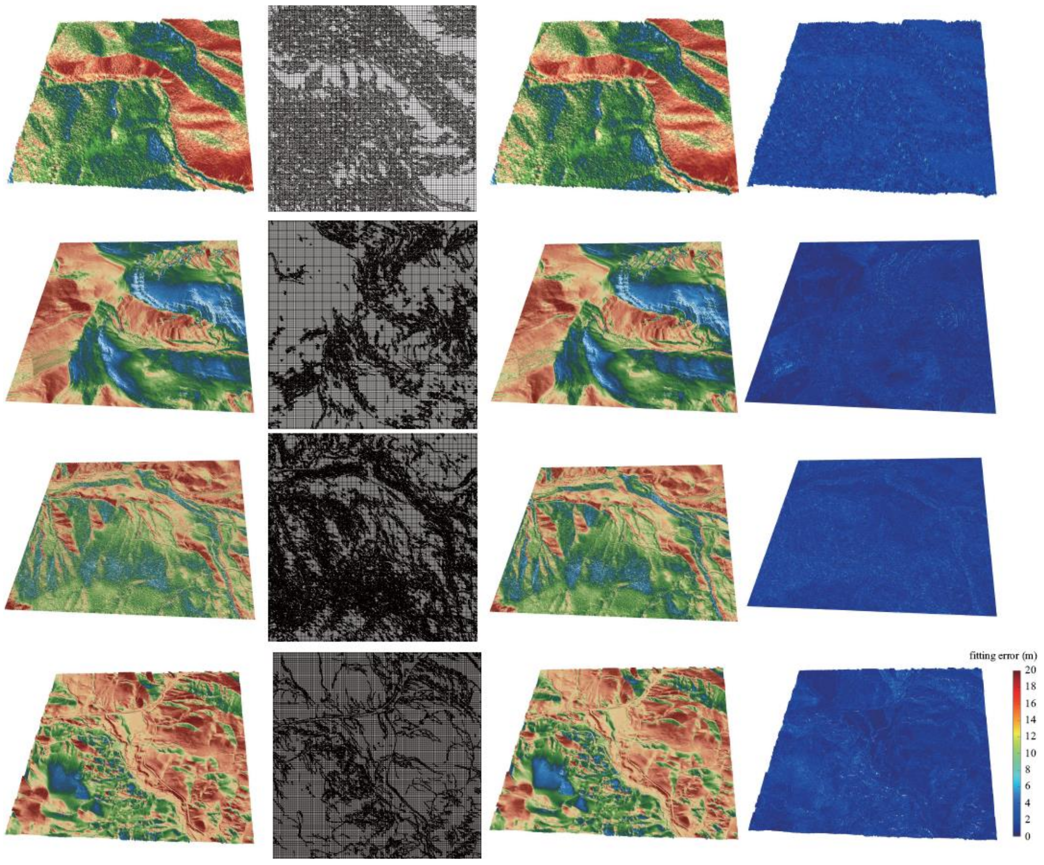

Figure 6: Fitting digital elevation model data. From left to right: raw data mapping, T-mesh, fitting data mapping, fitting error mapping.

5 CONCLUSION AND FUTURE WORK

In the paper, a fast T-spline fitting method is proposed based on feature extraction that can efficiently model large-size Z-map data such as high-resolution image data, digital elevation data, and so on. The edge detection technology is introduced from image processing to extract the feature of fitting data, and the ultimate T-mesh is constructed once based on the extracted feature without iteration. In addition to this, an efficient progressive iterative fitting method is presented with the help of T-spline local support ability. Two types of large-size Z-map data are tested to demonstrate the feasibility and efficiency of the proposed method comparing with the existing methods. The results prove the feature extraction is a high-efficiency tool for large-size Z-map data reconstruction using T-spline.

Edge detection is introduced for T-spline fitting from image processing, and the other feature extraction methods will be investigated for T-spline fitting. In the paper, the initial T-mesh is specified by the user to decides the number of the control points and the accuracy of the fitting results, an automatic initial T-mesh construction method will be considered in the future. Beyond that, a non-uniform initial T-mesh will be applied to improve efficiency further. Our proposed method can be re-implemented for volumetric data fitting naturally using trivariate T-spline, and the GPU based programming can be adopted to realize the real-time volumetric data fitting for biomedical analysis.

ACKNOWLEDGEMENTS

The work is supported by the Natural Science Foundation of China (Project No. 62102011).

Yazui Liu, http://orcid.org/0000-0002-7480-1535

REFERENCES

[1] Deng, J.-S.; Chen, F.-L.; Li, X.; Hu, C.-Q.; Tong, W.-H.; Yang, Z.-W.; Feng, Y.-Y.: Polynomial splines over hierarchical T-meshes, Graphical models, 70(4), 2008, 76-86. https://doi.org/10.1016/j.gmod.2008.03.001

[2] Feng, C.; Taguchi, Y.-C.: FasTFit: A fast T-spline fitting algorithm, Computer-Aided Design, 92, 2017, 11-21. https://doi.org/10.1016/j.cad.2017.07.002

[3] Floater, M.- S.; Hormann, K.: Surface parameterization: a tutorial and survey, Advances in multiresolution for geometric modelling. Springer, Berlin, Heidelberg, 2005: 157-186.

[4] Lee, S.-Y.; Wolberg, G.; Shin, S.-Y.: Scattered data interpolation with multilevel B-splines, IEEE transactions on visualization and computer graphics, 3(3), 1997, 228-244. https://doi.org/10.1109/2945.620490

[5] Li, X.; Deng, J.-S.; Chen, F.-L.: Polynomial splines over general T-meshes, The Visual Computer, 26(4), 2010, 277-286. https://doi.org/10.1007/s00371-009-0410-9

[6] Li, X.; Scott, M.-A: Analysis-suitable T-splines: characterization, refineability, and approximation, Mathematical Models and Methods in Applied Sciences, 24(06), 2014, 1141-1164. https://doi.org/10.1142/S0218202513500796

[7] Li, X.; Zhang, J.-J.: AS++ T-splines: Linear independence and approximation, Computer Methods in Applied Mechanics and Engineering, 333, 2018, 462-474. https://doi.org/10.1016/j.cma.2018.01.041

[8] Lin, H.-W.; Zhang, Z.-Y.: An efficient method for fitting large data sets using T-splines, SIAM Journal on Scientific Computing, 35(6), 2013, A3052-A3068. http://dx.doi.org/10.1137/120888569

[9] Lu, Z.-H.; Jiang, X.; Huo, G.-Y.; Ye, D.-L.; Wang, B.-L.; Zheng, Z.-M.: A fast T-spline fitting method based on efficient region segmentation, Computational and Applied Mathematics, 39(2), 2020, 55. https://doi.org/10.1007/s40314-020-1071-6

[10] Ma, W.; Zhao, N.: Smooth multiple B-spline surface fitting with Catmull-Clark subdivision surfaces for extraordinary corner patches, The Visual Computer, 18(7), 2002, 415-436. http://dx.doi.org/10.1007/s003710100159

[11] Nasri, A.; Sinno, K.; Zheng, J.-M.: Local T-spline surface skinning, The Visual Computer, 28(6-8), 2012, 787-797. https://doi.org/10.1007/s00371-012-0692-1

[12] Oh, M.-J.; Roh, M.-I.; Kim, T.-W.: Local T-spline surface skinning with shape preservation. Computer-Aided Design, 104, 2018, 15-26. https://doi.org/10.1016/j.cad.2018.04.006

[13] OpenTopography. High-Resolution Topography Data and Tools. https://opentopography.org/, 2022.

[14] Piegl, L.-A.; Tiller, W.: The NURBS book, Springer Science and Business Media, 2012. https://doi.org/10.1007/978-3-642-59223-2

[15] Sederberg, T.-W.; Zheng, J.-M.; Bakenov, A.; Nasri, A.: T-spline and T-NURCCs, ACM Transactions on Graphics, 22(3), 2003, 477-484. https://doi.org/10.1145/882262.882295

[16] Sederberg, T.-W.; Cardon, D.-L.; Finnigan, G.-T.; North, N.-S.; Zheng, J.-M.; Lyche, T.: T-spline simplification and local refinement, ACM Transactions on Graphics, 23(3), 2004, 276-283. https://dx.doi.org/10.1145/1015706.1015715

[17] Shang, C.; Fu, J.-Z.; Lin, Z.-W.; Feng, J.-W.; Li, B.: Closed T-Spline surface reconstruction from medical image data, International Journal of Precision Engineering and Manufacturing, 19(11), 2018, 1659-1671. https://doi.org/10.1007/s12541-018-0193-x

[18] Wang, Y.-M.; Zheng, J.-M.: Curvature-guided adaptive T-spline surface fitting, Computer-Aided Design, 45(8-9), 2013, 1095-1107. https://doi.org/10.1016/j.cad.2013.04.006

[19] Yang, H.-P.; Juttler, B.: Evolution of T-spline level sets for meshing non-uniformly sampled and incomplete data, The Visual Computer, 24(6), 2008, 435-448. https://doi.org/10.1007/s00371-008-0222-3

[20] Yang, X.-N.; Zheng, J.-M.: Approximate T-spline surface skinning, Computer-Aided Design, 44(12), 2012, 1269-1276. https://doi.org/10.1016/j.cad.2012.07.003

[21] Zheng, J.-M.; Wang, Y.-M.: Periodic t-splines and tubular surface fitting, International Conference on Curves and Surfaces. Springer, Berlin, Heidelberg, 2010: 731-746.

[22] Zheng, J.-M.; Wang, Y.-M.; Seah, H.-S.: Adaptive T-spline surface fitting to z-map models, Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia. 2005: 405-411.