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

Authors:
Yazui Liu, liuyazui@buaa.edu.cn, Beihang University
Gang Zhao, zhaog@buaa.edu.cn, Beihang University
Xiaoxiao Du, duxiaoxiao@buaa.edu.cn, Beihang University
Wei Wang, jrrt@buaa.edu.cn, Beihang University

Keywords:
T-spline, data fitting, feature extraction, Z-map data

DOI: 10.14733/cadconfP.2022.50-55

Introduction:
In recent years, the surface of the real objects can be easily measured or scanned with the development of the data capture devices. The measurement data represented mostly using a two-dimensional regular grid with height values is called Z-map data that is convenient for storage. However, Z-map data is a type of discrete representation data, 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, etc. 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.

T-spline has been widely adopted for complex data fitting with the advantages of fewer control points, local refinement, and watertight representation [1-3][5]. 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. Especially for the large-size Z-map data captured by the high-resolution device, which is time-consuming using the traditional two-phase T-spline fitting method.

In the 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 the T-spline local refinement without iteration, and an efficient progressive iterative fitting method is employed for T-spline control points evaluation. The computing costs can be reduced obviously since the proposed method is a single-phase iterative method. The proposed method is demonstrated using the high-resolution image data.

Main Idea:
T-spline
A T-spline surface is defined by a control grid called T-mesh, and is usually defined using the following equation[4]:

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

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

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

where $N_i(s)$ and $N_i(t)$are B-spline basis functions.

T-mesh construction based on feature extraction

The bicubic B-spline patch with $\max(\left\lfloor 20*\log_2^{\frac{m*n}{\theta}} \right\rfloor, 4) \times \max(\left\lfloor 20*\log_2^{\frac{m*n}{\theta}} \right\rfloor, 4)$ control grid is used as the initial T-mesh generally that can be used to fit flatness parts of the Z-map data, where m*n is the resolution of the Z-map data, $\lfloor x \rfloor$ denotes the integer number closest to x, and θ is specified by user to adjust the density of the initial T-mesh. To improve the fitting quality, additional control points need to be inserted over the steep parts, which have greater fitting error normally. In the paper, edge detection technology is introduced for Z-map data feature extraction due to its similar topology to image data. A huge amount of methods is investigated for edge detection, which uses first-order or second-order derivative typically. In the paper, Canny edge detection is employed 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. We denote Q_k is one of the candidate 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:

$$s_i \le s_k \le s_{i+1} \,\&\, t_j \le t_k \le t_{j+1} \tag{3}$$

A multi-split T-spline local refinement algorithm is introduced for all faces that have candidate inserted points as shown below:

Input: F_{old}
Output: F_{new}
step 1. Denote $Q_k, k\in(1,n)$ 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.
step 2.2. Remove F_k from F_{new}, and add F_{k1}, F_{k2} into F_{new}.

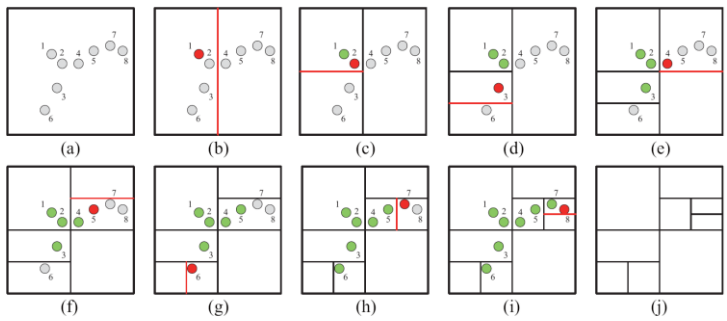

Fig. 1: An example of the multi-split T-spline local refinement algorithm. (a) presents one face with its candidate inserted points sorted by the fitting error, (b) - (i) presents the results of each step of the algorithm, the unprocessed points are marked in gray, the current selected point is marked 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 Fig 1, Fig 1(a) shows one of the faces with its candidate inserted points. Fig 1(b) – Fig 1(i) present the procedure of the multi-split T-spline local refinement algorithm. Fig 1(j) displays the final face after executing the multi-split T-spline local refinement algorithm.

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.

An efficient progressive iterative T-spline fitting method

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.

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

$$\begin{cases} S_i = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}] \\ T_i = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}] \end{cases} \tag{4}$$

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.

As shown in Fig 2. Fig 2(a) gives a T-mesh with its parametric space, the local knot vectors of control point P_1 are obtained intuitively as follows: $[s_2,s_3,s_4,s_6,s_7]$, $[t_1,t_2,t_3,t_5,t_6]$. Fig 2(b) shows the influence domain of T-spline control point P_1 defined by $(s_2,s_7)\otimes(t_1,t_6)$. Fig 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.

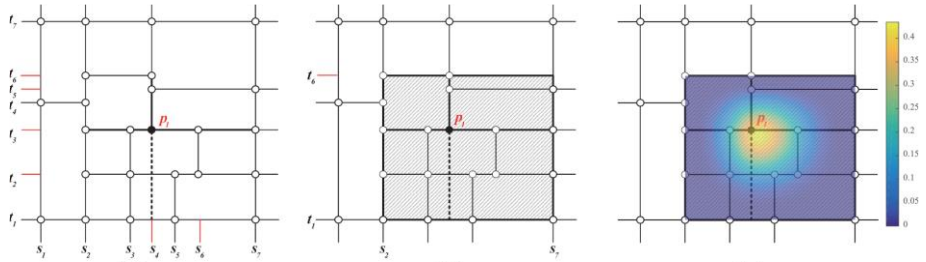

Fig. 2: (a) gives an example of T-mesh with its parametric space, (b) displays the influence domain of control point P_1, (c) presents the blending function of control point P_1.

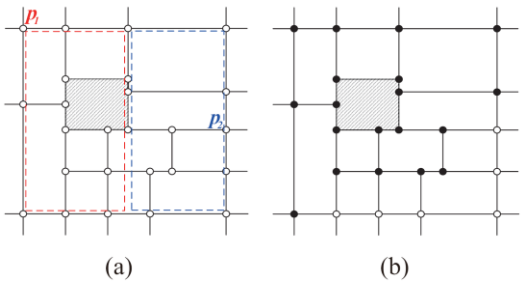

Fig. 3: (a) The relationship between the face and the influence domain of control points, (b) gives all face-influenced points of the shaded face marked by black solid circle.

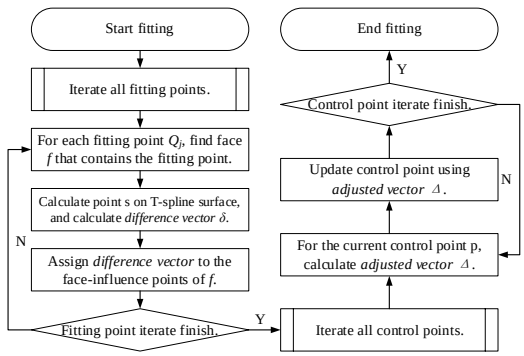

Fig. 4: The flow chart of the proposed efficient progressive iterative T-spline fitting method.

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.

As shown in Fig 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. Fig 3(b) shows all face-influenced points of the shaded face marked by black solid circle.

An efficient progressive iterative T-spline fitting method is presented based on face-influenced points to satisfy the need of large-size Z-map data fitting. The flow chart of the proposed method is shown in Fig 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 (3). 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 as below:

$$\begin{cases} P_i^{k+1} = P_i^k + \Delta_i^k \\ \Delta_i^k = \dfrac{\sum_{j=1}^{m} B_i(s_j,t_j)\delta_j^k}{\sum_{j=1}^{m} B_i(s_j,t_j)} \\ \delta_j^k = \overrightarrow{R^k(s_j,t_j)Q_j} \end{cases} \tag{5}$$

where P_i^{k+1} is the (k+1)st control point that is updated by *adjusted vector* Δ_i^k and P_i^k, Q_j is the fitting point, (s_j,t_j) is its parameter on the T-spline surface, $R^k(s_j,t_j)$ 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.

Experiments:

The 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 ultimate T-mesh construction based on Canny edge detection, and an efficient progressive iterative fitting method based on T-spline local support.

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 the proposed algorithm. In addition to this, Peak Signal to Noise Ration (PSNR) and Structural Similarity (SSIM) are employed to evaluate the quality of the fitting image. The fitting results are shown in Fig 5, and Tab 1 shows the statistic information.

Two images are used for comparison with reference [1] and [2], the comparison results are shown in Tab 2. For the two cases, our algorithm is faster than the other two methods. 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.

Conclusion:

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. 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. Four image 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.

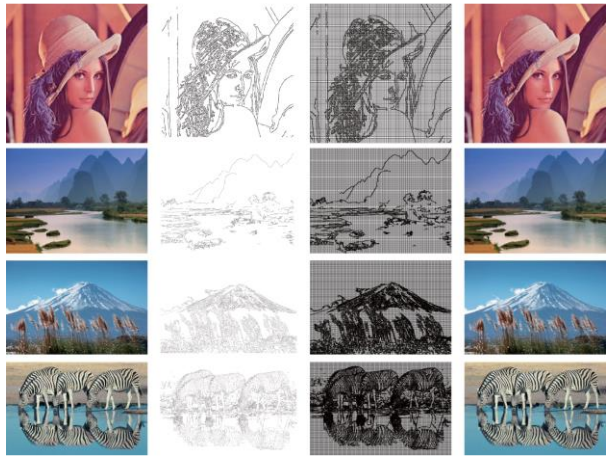

Fig. 5: Fitting high resolution image data. From left to right: raw image, feature extraction results, T-mesh, fitting image.

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

Tab. 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

Tab. 2: Comparison between our algorithm, lin2013[2] and Feng2017[1] for image fitting.

Acknowledgements:
This research was supported by the Chinese National Natural Science Foundation through grants 62102011.

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

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

[3] Lu, Z.; Jiang, X.; Huo, G.; Ye, D.; Wang, B.; Zheng, J.: 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

[4] Sederberg, T.-W.; Cardon, L.-D.; Finnigan, T.-G.; North, S.-N.; Zheng, J; Lyche, T.: T-splines simplification and local refinement, ACM transactions on graphics (TOG), 23(3), 2004, 276-283. https://doi.org/10.1145/1015706.1015715

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