SIAM J. SCI. COMPUT.
Vol. 35, No. 6, pp. A3052–A3068

AN EFFICIENT METHOD FOR FITTING LARGE DATA SETS USING T-SPLINES[*]

HONGWEI LIN[†] AND ZHIYU ZHANG[†]

Abstract. Data fitting is a fundamental tool in scientific research and engineering applications. Generally, there are two ingredients in solving data fitting problems. One is the fitting representation, and the other is the fitting method. Nowadays, the fitting of larger and larger quantities of data sets requires more compact fitting representation and faster fitting methods. The T-spline is a recently invented spline representation, whose control mesh (*T-mesh*) allows a row of control points to terminate, thus reducing the number of superfluous control points in the B-spline representation significantly. This property makes T-splines more compact than B-splines in fitting large data sets. However, the adaptivity of the T-spline causes the coefficient matrix of a least-squares fitting linear system to lose its block structure. Thus, when fitting large data sets with T-splines by iterative methods, only point iterative methods can be used, and the iteration speeds of typically employed point iterative methods are rapidly slowed with the increasing number of unknowns. In this paper, we present a *progressive T-spline data fitting* algorithm for fitting large data sets with a T-spline representation. As an iterative method, the iteration speed of our method is steady and insensitive to the growing number of unknown T-mesh vertices; thus, it is able to fit large data sets efficiently. Additionally, our method can handle data sets with or without holes in a unified framework, without any special processing. Finally, we apply the progressive T-spline data fitting algorithm in large-image fitting to validate its efficiency and effectiveness.

Key words. data fitting, iterative method, T-splines, image fitting

AMS subject classifications. 65D07, 65D10, 65D17, 65D18

DOI. 10.1137/120888569

1. Introduction.

Data fitting is a fundamental tool in scientific research and engineering applications. With the development of the data capture devices, real objects or models can be easily measured or scanned, outputting highly precise and regular measurement data in larger and larger quantities [2]. This naturally raises the problems of how to fit these larger and larger data sets efficiently, and how to make the fitting representation as compact as possible.

Actually, there are two ingredients in solving the data fitting problem. One is the fitting method; the other is the fitting representation. When the data sets are very large, the linear systems that result from data fitting are too large to fit in the main memory-storage systems of usual devices. Therefore, direct-solution techniques such as Cholesky decomposition become impractical, and iteration methods are usually employed in fitting large data sets.

On the other hand, spline functions, especially B-splines, are commonly taken as the representation in fitting large data sets, because spline functions can avoid the Runge phenomenon appearing in data fitting with a high degree of polynomials. Nevertheless, the control net of the B-spline surface is a regular grid, so it requires many superfluous control points simply to satisfy the topological structure constraints of the control net, especially in fitting large data sets. To overcome the topological

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section August 20, 2012; accepted for publication (in revised form) October 9, 2013; published electronically December 19, 2013. This research was supported by the Natural Science Foundation of China (61379072, 60933008, 60970150).
http://www.siam.org/journals/sisc/35-6/88856.html

[†]Department of Mathematics, State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, China (hwlin@zju.edu.cn, zhangzhiyu@zjucadcg.cn).

constraints of the B-spline control net, the T-spline [13, 14] is invented, which is a generalization of the B-spline. The T-spline control mesh (called *T-mesh*) allows a row of control points to terminate, forming a T-junction, such that it is able to significantly reduce the number of superfluous control points in the B-spline representation, making the fitting representation much more compact. Therefore, the T-spline is a desirable representation for fitting large data sets.

However, unlike B-spline least-squares fitting, in which the coefficient matrix of a linear system is block structured and block iterative methods [19, 12] can be applied to accelerate the computation speed, the adaptivity of the T-spline causes the coefficient matrix of a least-squares fitting linear system to lose its block structure. Therefore, when fitting large data sets with T-splines by iterative methods, only point iterative methods [19, 12] can be used. Because the iteration speed of typically employed point iterative methods, such as the Gauss–Seidel and conjugate gradient methods, is rapidly slowed with the increasing number of unknowns, they are inefficient when fitting large data sets.

To overcome the difficulties stated above, in this paper, we develop a *progressive T-spline data fitting* algorithm for fitting large data sets, which consists of two alternately executed procedures, *progressive fitting* and *subregional knot insertion*. Starting with an initial T-mesh, a progressive fitting algorithm is proposed to fit the initial T-spline to the data set; after the progressive fitting terminates, a subregional knot-insertion procedure is invoked to insert knots into the current T-spline. Thus, the number of T-mesh vertices is gradually increased. The two procedures are performed alternately until a prescribed precision is reached. Compared with existing methods, our method has the following advantages:

1. The iteration speed of the progressive fitting algorithm is steady and insensitive to the increasing number of unknown T-mesh vertices, due to the local support property of T-spline blending functions; thus, this algorithm is able to efficiently fit large data sets.
2. Because of the adaptivity of T-spline representation, our method can fit the large data sets adaptively and substantially reduce the number of control mesh vertices.

In addition, when there are holes in the large data sets, the coefficient matrix of the linear system will have small singular values and be ill conditioned [10]. So, traditional methods should be elaborately modified to deal with large data sets with holes [10]. However, as another advantage of our progressive fitting algorithm, it can handle data sets with or without holes in a unified framework, without special processing.

Moreover, to validate the efficiency and effectiveness of our method, we apply our method to large-image fitting and compare it with traditional iterative methods and B-spline fitting results.

The structure of this paper is as follows. We first briefly review related work in section 1.1 and then introduce the T-spline representation in section 1.2 to make this paper self-contained. In section 2, we present the entire algorithm in detail. After presenting some results and discussions in section 3, we conclude the paper in section 4.

1.1. Related work.

Data fitting with splines has been widely applied in science and engineering, and there is much literature on this topic [5, 4]. Iterative methods are also widely employed in solving large sparse linear systems [19, 12] and other related areas, such as reconstruction of original signals from their samples [1]. Since

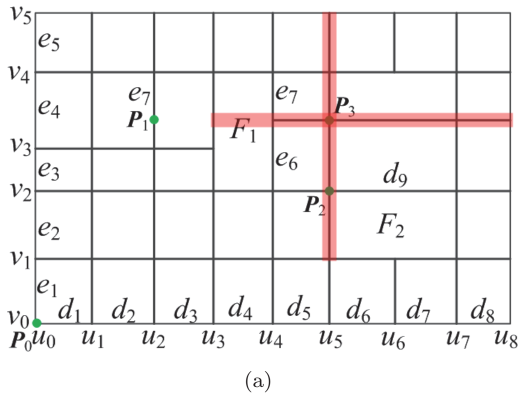
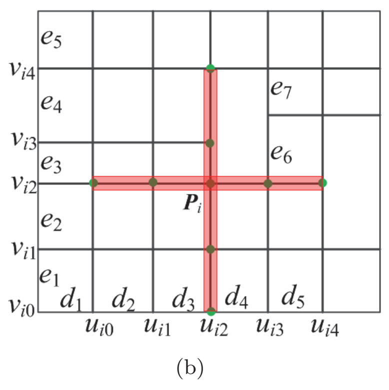

v_5
v_4
v_3
v_2
v_1
v_0
e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_7
P_0 P_1 P_2 P_3
F_1 F_2
d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9
u_0 u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8
(a)

v_{i4}
v_{i3}
v_{i2}
v_{i1}
v_{i0}
e_1 e_2 e_3 e_4 e_5 e_6 e_7
P_i
d_1 d_2 d_3 d_4 d_5
u_{i0} u_{i1} u_{i2} u_{i3} u_{i4}
(b)

FIG. 1.1. (a) *Preimage of a T-mesh.* (b) *Knot lines, u-edges, and v-edges (red shading) for the blending function* $B_i(u, v)$ (1.2).

we take the problem of image fitting as an example to validate the efficiency and effectiveness of our method, we focus on the work related to image fitting with splines. As pointed out in [17], the spline is a perfect fit for signal and image processing. After spline fitting was introduced to image processing [7], it has been successfully applied in image zooming [18], geometric image transformation [16], image reduction [18, 8], image compression [9], etc. However, in these applications, B-splines are usually taken as the fitting representation. Because many superfluous control points are required to just satisfy the topological structure constraints of the B-spline control net, it makes the B-spline representation bulky when fitting large data sets.

As a generalization of the B-spline, the T-spline [13, 14] control mesh allows a row of control points to terminate, thus greatly reducing the number of superfluous control points for topological constraints of the B-spline control net. Moreover, Wang and Zheng developed the control-point-removal algorithm for T-splines [20]. Buffa, Cho, and Sangalli studied the linear independence of the T-spline blending functions associated with several particular T-meshes [3]. Inspired by T-splines, He et al. developed the manifold T-spline, which is defined over an arbitrary manifold domain of any topological type [6]. T-splines have been applied in various areas of study. Song and Yang constructed a T-spline volume for deformation [15]. Zheng, Wang, and Seah used T-splines to fit Z-map models [21]. However, their algorithm is impractical in fitting large-image data sets.

1.2. A brief introduction to T-splines. In this section, we will briefly introduce the representation of a T-spline. More details can be found in [13]. The control grid of a T-spline patch is called a *T-mesh*. Figure 1.1(a) illustrates the preimage of a T-mesh in the (u, v) parameter space. In the preimage, the knot intervals d_i and e_i are nonnegative numbers indicating the differences between two knots, and these are assigned to edges of the T-mesh. A valid T-mesh requires the sum of all knot intervals along one side of any face to equal the sum of the knot intervals on the opposite side. For instance, in Figure 1.1, $e_3 + e_4 = e_6 + e_7$ on face F_1, and $d_6 + d_7 = d_9$ on face F_2.

If we designate the origin $(0, 0)$ of the (u, v) parameter space, a local knot coordinate system can be inferred from the knot intervals of a T-mesh. In this case, each control point in the T-mesh possesses its knot coordinates. For example, in Figure

1.1, if we choose $(u_0, v_0) = (0, 0)$, then $u_1 = d_1$, $u_2 = d_1 + d_2$, $v_1 = e_1$, $v_2 = e_1 + e_2$, and so on. Therefore, the knot coordinates for $\boldsymbol{P}_0$ are $(0, 0)$, for $\boldsymbol{P}_1$ are $(u_2, v_2 + e_6)$, for $\boldsymbol{P}_2$ are (u_5, v_2), and for $\boldsymbol{P}_3$ are $(u_5, v_2 + e_6)$.

A T-spline is a point-based spline. Each of its control points $\boldsymbol{P}_i,\ i = 0, 1, \ldots, n$, corresponds to a blending function,

$$B_i(u, v) = \frac{w_i M_i(u, v)}{\sum_{j=0}^{n} w_j M_j(u, v)}, \quad i = 0, 1, \ldots, n, \tag{1.1}$$

where w_i are nonnegative weights, and

$$M_i(u, v) = N[u_{i0}, u_{i1}, u_{i2}, u_{i3}, u_{i4}](u) N[v_{i0}, v_{i1}, v_{i2}, v_{i3}, v_{i4}](v). \tag{1.2}$$

In (1.2), $N[u_{i0}, u_{i1}, u_{i2}, u_{i3}, u_{i4}](u)$ is a cubic B-spline basis function defined on the u-directional knot vector,

$$\boldsymbol{u}_i = [u_{i0}, u_{i1}, u_{i2}, u_{i3}, u_{i4}], \tag{1.3}$$

and $N[v_{i0}, v_{i1}, v_{i2}, v_{i3}, v_{i4}](v)$ is another cubic B-spline basis function defined on the v-directional knot vector,

$$\boldsymbol{v}_i = [v_{i0}, v_{i1}, v_{i2}, v_{i3}, v_{i4}]. \tag{1.4}$$

The knot vectors $\boldsymbol{u}_i$ (1.3) and $\boldsymbol{v}_i$ (1.4) are inferred from the T-mesh neighborhood of $\boldsymbol{P}_i$ by the following rule [13] (Figure 1.1(b)).

Rule 1. (u_{i2}, v_{i2}) are the knot coordinates of $\boldsymbol{P}_i$ (see Figure 1.1(b)). Consider a ray in the parameter space, $\boldsymbol{R}(\alpha) = (u_{i2} + \alpha, v_{i2})$. Then, u_{i3} and u_{i4} are the u coordinates of the first two u-edges intersected by the ray (not including the initial point (u_{i2}, v_{i2})). By u-edge we mean a vertical line segment of constant u (refer to Figure 1.1(b)); similarly, a v-edge is a horizontal line segment of constant v. The other knots in $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are found in a similar manner.

For instance, in Figure 1.1(a), the u-directional knot vector for $\boldsymbol{P}_3(u_5, v_2 + e_6)$ is $[u_3, u_4, u_5, u_7, u_8]$, and the v-directional knot vector for it is $[v_1, v_2, v_2 + e_6, v_4, v_5]$.

In our implementation, we let $w_i = 1$, $i = 0, 1, \ldots, n$. After getting the blending function $B_i(u, v)$ corresponding to each control point $\boldsymbol{P}_i$, a T-spline patch $\boldsymbol{T}(u, v)$ can be generated by blending the control points $\boldsymbol{P}_i$ and the functions $B_i(u, v)$, $i = 0, 1, \ldots, n$, i.e.,

$$\boldsymbol{T}(u, v) = \sum_{i=0}^{n} \boldsymbol{P}_i B_i(u, v).$$

2. The algorithm.

2.1. Overview. The progressive T-spline data fitting algorithm starts with an initial cubic B-spline patch. Then, a progressive method is developed to make the patch approximate the input data set. When the progressive fitting iteration reaches its termination condition, a well-designed subregional knot-insertion strategy is employed to insert knots with large fitting errors to the patch, and a new progressive fitting procedure is invoked. The two procedures are performed alternately until a prescribed precision is attained. The entire algorithm is summarized in Algorithm 1, the details of which will be elucidated in the following sections.

Algorithm 1: Progressive T-spline Data Fitting.

```
    Input: A data set and a predefined fitting precision $\varepsilon_0$
    Output: A T-spline mesh
1   Data parametrization and initial B-spline patch construction ;
2   Data fitting by the progressive method ;
3   while The current fitting error $\varepsilon > \varepsilon_0$ do
4       Insert knots to the current patch ;
5       Data fitting by the progressive method ;
6   end
```

2.2. Parametrization and initial patch construction. Suppose the given data set $\mathcal{M}$ is arranged as a $\varphi \times \psi$ array and the data point at the position (w, h) is

$$\boldsymbol{Q}_{wh} = (x_{wh}, y_{wh}, z_{wh}), \quad w = 0, 1, \ldots, \varphi - 1, \quad h = 0, 1, \ldots, \psi - 1.$$

To fit these points with a T-spline patch, each of them should be assigned a pair of parameters (u_w, v_h), where $u_0 < u_1 < \cdots < u_{\varphi-1}$, $v_0 < v_1 < \cdots < v_{\psi-1}$ is called *parametrization*.

Generally, there are two kinds of most often employed parametrization methods. One is the normalized accumulated chord length method [11], and the other is the uniform parametrization method. In geometric design and related fields, the parameters for the given data points are usually generated by the former method. Specifically, we first calculate the parameters $(u_0^h, u_1^h, \ldots, u_{\varphi-1}^h)$ for each row of data points $(\boldsymbol{Q}_{0h}, \boldsymbol{Q}_{1h}, \ldots, \boldsymbol{Q}_{\varphi-1,h})$, $h = 0, 1, \ldots, \psi - 1$, using the normalized accumulated chord length method [11], i.e.,

$$u_0^h = 0, \quad u_w^h = \frac{\sum_{j=0}^{w-1} \|\boldsymbol{Q}_{j+1,h} - \boldsymbol{Q}_{j,h}\|}{\sum_{j=0}^{\varphi-2} \|\boldsymbol{Q}_{j+1,h} - \boldsymbol{Q}_{j,h}\|}, \quad w = 1, 2, \ldots, \varphi - 1,$$

where $\|\boldsymbol{Q}_{j+1,h} - \boldsymbol{Q}_{j,h}\|$ denotes the length of the line segment $\boldsymbol{Q}_{j,h}\boldsymbol{Q}_{j+1,h}$. Then, the u-parameters are generated by averaging each column of parameters, i.e.,

$$u_w = \frac{\sum_{h=0}^{\psi-1} u_w^h}{\psi}, \quad w = 0, 1, \ldots, \varphi - 1.$$

Moreover, the v-parameters v_h, $h = 0, 1, \ldots, \psi - 1$, can be calculated similarly.

However, in image fitting, the rgb-color values (r_{wh}, g_{wh}, b_{wh}) of each pixel are taken as the coordinates (x_{wh}, y_{wh}, z_{wh}) of data points, and their parameters (u_w, v_h) are usually assigned as the indices (w, h) of the pixel, i.e., $(u_w, v_h) = (w, h)$. It is actually the uniform parametrization method.

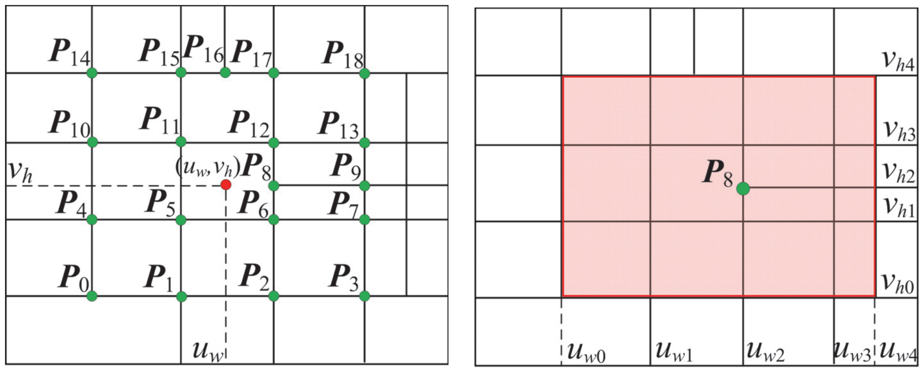
Additionally, the initial patch has an influence over the configuration of the final T-mesh. While the initial patch with too dense control points will make the final T-mesh contain many superfluous vertices, the one with too sparse control points will prolong the iteration time. Then, to balance the number of final T-mesh vertices and the iteration time, in this paper, we take a bicubic B-spline patch with $\max\{\lfloor \frac{\varphi}{100} \rfloor, 4\} \times \max\{\lfloor \frac{\psi}{100} \rfloor, 4\}$ uniformly distributed control points as the initial patch, where $\lfloor \frac{\varphi}{100} \rfloor$ denotes the smallest integer larger than or equal to $\frac{\varphi}{100}$. Although the control points of the initial patch are uniformly distributed, the subregional knot-insertion strategy will insert knots adaptively, allowing the T-spline to capture the features of the given data set automatically.

2.3. T-spline progressive fitting and its convergence. As stated in section 1.1, because the coefficient matrix of a T-spline least-squares fitting system loses its block structure, only point iterative methods are available to solve the T-spline least-squares fitting problem. In this section, we develop an efficient progressive algorithm for fitting a T-spline to a large data set.

After constructing the initial patch or inserting knots into the current T-spline, a T-spline progressive fitting procedure is invoked to make the T-spline patch approximate the data set. The progressive fitting algorithm is an iterative method that begins with an initial patch $\boldsymbol{T}^{(0)}(u,v)$. We take the kth step of the iterations as an example to explain the algorithm. Suppose the T-spline patch after the $(k-1)$st iteration is $\boldsymbol{T}^{(k)}(u,v)$ with T-mesh vertices $\boldsymbol{P}_i^{(k)}$, $i=0,1,\ldots,n$, i.e.,

$$\boldsymbol{T}^{(k)}(u,v)=\sum_{i=0}^{n}\boldsymbol{P}_i^{(k)}B_i(u,v), \tag{2.1}$$

where $B_i(u,v)$ is the T-spline blending function. As noted in [13], a T-spline is point-based, where each T-mesh vertex $\boldsymbol{P}_i^{(k)}$ corresponds to a blending function $B_i(u,v)$.



(a) Vector distribution. (b) Vector gathering.

FIG. 2.1. (a) *The point at* (u_w,v_h) *of a T-spline patch is a linear combination of the T-mesh vertices* $\boldsymbol{P}_i$, $i=0,1,\ldots,18$; *therefore, the difference vector* $\boldsymbol{\delta}_{wh}^k$ *at* (w,h) *distributes* $B_i(u_w,v_h)\boldsymbol{\delta}_{wh}^k$ *to the vertices* $\boldsymbol{P}_i$. (b) *The blending function* $B_8(u,v)$ *corresponding to the T-mesh vertex* $\boldsymbol{P}_8$ *is nonzero in the region* $(u_{w_0},u_{w_4})\times(v_{h_0},v_{h_4})$, *and all of the pixels in this region distribute their weighted difference vectors to the vertex* $\boldsymbol{P}_8$. *Therefore, the difference vector* $\boldsymbol{\Delta}_8^k$ *for* $\boldsymbol{P}_8$ *is generated by gathering these weighted vectors.*

Because of the localization of the blending function, the value $\boldsymbol{T}^{(k)}(u_w,v_h)$ is a linear combination of several T-mesh vertices $\boldsymbol{P}_{h_0}^{(k)},\boldsymbol{P}_{h_1}^{(k)},\ldots,\boldsymbol{P}_{h_l}^{(k)}$, whose blending function is nonzero at (u_w,v_h) (see Figure 2.1(a)),

$$\boldsymbol{T}^{(k)}(u_w,v_h)=\boldsymbol{P}_{h_0}^{(k)}B_{h_0}(u_w,v_h)+\boldsymbol{P}_{h_1}^{(k)}B_{h_1}(u_w,v_h)+\cdots+\boldsymbol{P}_{h_l}^{(k)}B_{h_l}(u_w,v_h).$$

To generate the $(k+1)$st T-spline patch $\boldsymbol{T}^{(k+1)}(u,v)$, we first calculate the *difference vector for each data point* $\boldsymbol{Q}_{wh}$, i.e.,

$$\boldsymbol{\delta}_{wh}^{(k)}=\boldsymbol{Q}_{wh}-\boldsymbol{T}^{(k)}(u_w,v_h), \tag{2.2}$$

and distribute the weighted vector $B_{h_\alpha}(u_w,v_h)\boldsymbol{\delta}_{wh}^{(k)}$ to the T-mesh vertex $\boldsymbol{P}_{h_\alpha}^{(k)}$, $\alpha=0,1,\ldots,l$ (Figure 2.1(a)).

Next, all of the weighted vectors distributed to the T-mesh vertex $\boldsymbol{P}_i^{(k)}$ are gathered to generate the *difference vector* $\boldsymbol{\Delta}_i^{(k)}$ *for the T-mesh vertex* $\boldsymbol{P}_i^{(k)}$ in the following manner (see Figure 2.1(b)):

$$\boldsymbol{\Delta}_i^{(k)} = \frac{\sum_{(w,h)\in I_i} B_i(u_w, v_h)\boldsymbol{\delta}_{w,h}^{(k)}}{\sum_{(w,h)\in I_i} B_i(u_w, v_h)}, \tag{2.3}$$

where I_i is the index set of the pixels that distribute their difference vectors to the control point $\boldsymbol{P}_i^{(k)}$. In fact, I_i contains the coordinates (w,h) of the data points fulfilling the condition $B_i(u_w, v_h) \neq 0$. As illustrated in Figure 2.1(b), the blending function $B_8(u,v)$ corresponding to the T-mesh vertex $\boldsymbol{P}_8$ is nonzero in the region $(u_{w_0}, u_{w_4}) \times (v_{h_0}, v_{h_4})$; thus, the difference vector $\boldsymbol{\Delta}_8^k$ for $\boldsymbol{P}_8$ is generated by gathering these weighted vectors, i.e.,

$$\boldsymbol{\Delta}_8^{(k)} = \frac{\sum_{w=w_0+1}^{w_4-1}\sum_{h=h_0+1}^{h_4-1} B_8(u_w, v_h)\boldsymbol{\delta}_{wh}^{(k)}}{\sum_{w=w_0+1}^{w_4-1}\sum_{h=h_0+1}^{h_4-1} B_8(u_w, v_h)}.$$

Therefore, the T-mesh vertices for the (k + 1)st T-spline patch $\boldsymbol{T}^{(k+1)}(u,v)$ are produced by

$$\boldsymbol{P}_i^{(k+1)} = \boldsymbol{P}_i^{(k)} + \boldsymbol{\Delta}_i^{(k)}, \quad i = 0, 1, \ldots, n, \tag{2.4}$$

and we obtain the new T-spline patch $\boldsymbol{T}^{(k+1)}(u,v)$ for the (k + 1)st iteration.

Convergence. Arranging the difference vectors for T-mesh vertices in a sequence,

$$\Delta^{(k)} = [\boldsymbol{\Delta}_1^{(k)}, \boldsymbol{\Delta}_2^{(k)}, \ldots, \boldsymbol{\Delta}_n^{(k)}]^T,$$

we can use (2.2) and (2.3) to represent the iterative format in matrix form,

$$\Delta^{(k+1)} = (E - \Lambda A^T A)\Delta^{(k)}, \tag{2.5}$$

where E is an identity matrix, Λ is the diagonal matrix,

$$\Lambda = \operatorname{diag}\left\{\frac{1}{\sum_{(w,h)\in I_0} B_0(u_w, v_h)}, \frac{1}{\sum_{(w,h)\in I_1} B_1(u_w, v_h)}, \ldots, \frac{1}{\sum_{(w,h)\in I_n} B_n(u_w, v_h)}\right\},$$

and

$$A = \begin{bmatrix} B_0(u_0, v_0) & B_1(u_0, v_0) & \cdots & B_n(u_0, v_0) \\ B_0(u_1, v_0) & B_1(u_1, v_0) & \cdots & B_n(u_1, v_0) \\ \cdots & \cdots & \cdots & \cdots \\ B_0(u_{\varphi-1}, v_0) & B_1(u_{\varphi-1}, v_0) & \cdots & B_n(u_{\varphi-1}, v_0) \\ B_0(u_0, v_1) & B_1(u_0, v_1) & \cdots & B_n(u_0, v_1) \\ \cdots & \cdots & \cdots & \cdots \\ B_0(u_{\varphi-1}, v_1) & B_1(u_{\varphi-1}, v_1) & \cdots & B_n(u_{\varphi-1}, v_1) \\ \cdots & \cdots & \cdots & \cdots \\ B_0(u_{\varphi-1}, v_{\psi-1}) & B_1(u_{\varphi-1}, v_{\psi-1}) & \cdots & B_n(u_{\varphi-1}, v_{\psi-1}) \end{bmatrix}.$$

On the one hand, A^TA is a positive definite matrix if it is nonsingular. Because the nonsingularity of the matrix A^TA follows the linear independence of the T-spline blending functions (1.1), which was established by [3] for several particular T-meshes, we make the assumption that it holds for our T-mesh.

Therefore, all eigenvalues of A^TA, as well as those of ΛA^TA, are positive, i.e., $\lambda(\Lambda A^TA) > 0$. Indeed, if λ is an eigenvalue and v is a nonzero eigenvector, then $\lambda = v^TA^TAv/v^T\Lambda^{-1}v > 0$, because both A^TA and Λ^{-1} are positive definite matrices.

On the other hand, $\|A\|_\infty \leq 1$, because the T-spline blending functions (1.1) are nonnegative and form a partition of unity, and thus $\|\Lambda A^T\|_\infty \leq 1$, as this matrix corresponds to the averaging procedure that defines the difference vectors in (2.3). Therefore, $\|\Lambda A^TA\|_\infty \leq \|A\|_\infty \|\Lambda A^T\|_\infty \leq 1$, and then all of its eigenvalues are less than or equal to 1, i.e., $\lambda(\Lambda A^TA) \leq 1$.

Consequently, the eigenvalues of the matrix $E - \Lambda A^TA$ satisfy $0 \leq \lambda(E - \Lambda A^TA) < 1$. This result means that the progressive fitting iteration format (2.5) is convergent.

In fact, the limit of the progressive fitting format is the least-squares fitting result for the data set. Let

$$\begin{aligned} P^{(k)} &= [\boldsymbol{P}_0^{(k)}, \boldsymbol{P}_2^{(k)}, \ldots, \boldsymbol{P}_n^{(k)}]^T, \\ Q &= [\boldsymbol{Q}_{00}, \boldsymbol{Q}_{10}, \ldots, \boldsymbol{Q}_{\varphi-1,0}, \ldots, \boldsymbol{Q}_{01}, \ldots, \boldsymbol{Q}_{\varphi-1,1}, \ldots, \boldsymbol{Q}_{\varphi-1,\psi-1}]^T. \end{aligned}$$

Based on (2.2) and (2.3), we can see that

$$\Delta^{(k)} = \Lambda A^T(Q - AP^{(k)}).$$

Together with (2.4), we have

$$\begin{aligned} P^{(k+1)} &= P^{(k)} + \Delta^{(k)} = P^{(k)} + \Lambda A^T(Q - AP^{(k)}) \\ &= (E - \Lambda A^TA)P^{(k)} + \Lambda A^TQ \\ &= (E - \Lambda A^TA)^{k+1}P^{(0)} + \sum_{l=0}^{k}(E - \Lambda A^TA)^l \Lambda A^TQ. \end{aligned}$$

Because $0 \leq \lambda(E - \Lambda A^TA) < 1$, we have

$$\lim_{k\to\infty}(E - \Lambda A^TA)^k = 0 \quad \text{and} \quad \sum_{l=0}^{\infty}(E - \Lambda A^TA)^l = (\Lambda A^TA)^{-1}.$$

Therefore, when $k \to \infty$,

$$P^{(\infty)} = (\Lambda A^TA)^{-1}\Lambda A^TQ.$$

That is,

$$\Lambda A^TAP^{(\infty)} = \Lambda A^TQ, \text{ equivalent to } A^TAP^{(\infty)} = A^TQ.$$

This is the normal equation of the least-squares fitting system, so the limit of the progressive fitting iterations is the least-squares fitting solution to the data points $\boldsymbol{Q}_{wh}$, $w = 0, 1, \ldots, \varphi - 1$, $h = 0, 1, \ldots, \psi - 1$.

The progressive fitting iterations are terminated when either $|\frac{\varepsilon^{(k-1)}}{\varepsilon^{(k)}} - 1| < \eta$ or the iteration number exceeds a prescribed threshold, N. The fitting error $\varepsilon^{(k)}$ is defined as the root mean square (RMS)

$$\varepsilon^{(k)} = \sqrt{\frac{\sum_{w=0}^{\varphi-1}\sum_{h=0}^{\psi-1}\|\boldsymbol{T}^{(k)}(u_w, v_h) - \boldsymbol{Q}_{wh}\|^2}{\varphi\psi}}. \tag{2.6}$$

In our image-fitting implementation, we take $\eta = 5 \times 10^{-5}$, and $N = 50$.

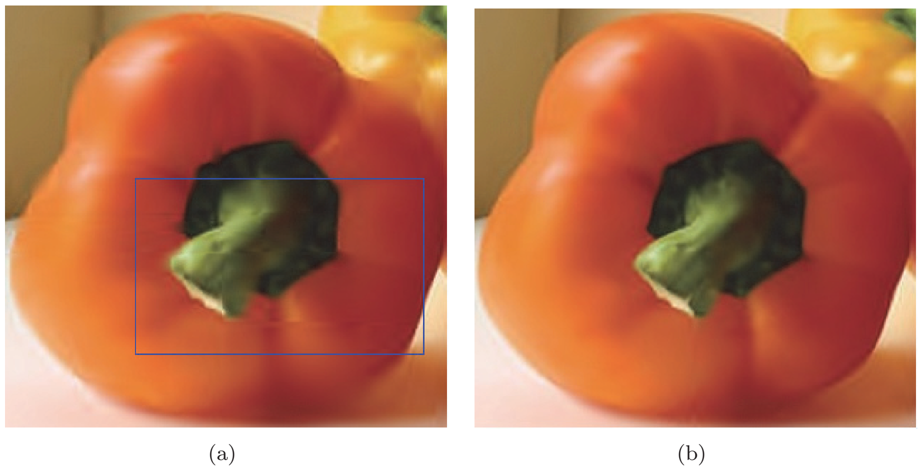


FIG. 2.2. *Comparison between two knot-insertion methods.* (a) *Part of the reconstructed image from the T-spline generated by inserting knots at pixels with largest errors, with PSNR* 38.5319 *dB.* (b) *Part of the reconstructed image from the T-spline generated by subregional knot-insertion method, with PSNR* 38.8534 *dB.*

2.4. Subregional knot insertion. After the entire progressive fitting procedure is terminated and if the fitting precision ε (2.6) of the current T-spline does not reach the prescribed precision ε_0 (refer to Algorithm 1), we insert several knots into the current T-spline to improve its degree of freedom.

Denoting the current T-spline patch as $\boldsymbol{T}(u,v)$, a natural choice for knot insertion is to select and insert the knots at the data points with the largest fitting errors,

$$e_{wh} = \|\boldsymbol{T}(u_w, v_h) - \boldsymbol{Q}_{wh}\|. \tag{2.7}$$

However, in many cases, the data points with the largest errors are very close to each other. As a result, the inserted knots and knot lines are also very close to each other, leading to unwanted wiggles in the resulting T-spline patch (refer to Figure 2.2(a)).

To avoid wiggles, we develop a *subregional knot-insertion* method. First, the entire data point array $\mathcal{M}$ is uniformly segmented into $m_0 \times n_0$ subregions. In our implementation, we take

$$m_0 = \lfloor \tfrac{\varphi}{100} \rfloor \quad \text{and} \quad n_0 = \lfloor \tfrac{\psi}{100} \rfloor.$$

After the Kth round progressive fitting procedure terminates, the entire image is further uniformly subdivided into $Km_0 \times Kn_0$ subregions, where $K = 1, 2, \ldots$. Then, we choose $\alpha\%$ subregions, i.e., $\lfloor (Km_0 \times Kn_0)\frac{\alpha}{100} \rfloor$ subregions, which have the largest RMS errors (2.6). Here, $\alpha\%$ is called the *insertion ratio.* Moreover, in each subregion that was subsequently selected, we search for the data point $\boldsymbol{Q}_{wh}$ with the largest fitting error e_{wh} (2.7). Suppose the knot vectors of the blending function at (u_w, v_h) are $[u_0, u_1, u_2, u_3, u_4]$ and $[v_0, v_1, v_2, v_3, v_4]$, where $u_2 = u_w$, $v_2 = v_h$ (which can be generated by Rule 1 in [13]); we insert a knot at $(\frac{u_1+u_3}{2}, \frac{v_1+v_3}{2})$.

After all of the $\lfloor (Km_0 \times Kn_0)\frac{\alpha}{100} \rfloor$ knots are inserted, the $(K+1)$st round progressive fitting procedure is initiated. The subregional knot-insertion method makes the resulting T-spline patch very smooth (see Figure 2.2(b)).

It should be pointed out that the insertion ratio $\alpha\%$ will affect the number of final T-mesh vertices and iteration time. For a predefined fitting precision, if the insertion ratio $\alpha\%$ becomes larger, the iteration time will be shorter, but the final T-mesh will include more vertices; in contrast, if it is smaller, the iteration time will be made longer, and the final T-mesh will contain fewer vertices. The effect of the insertion ratio $\alpha\%$ will be further explained in section 3.

2.5. Computation of the T-spline. As stated in section 2.3, in each step of the progressive fitting iterations, the values $\boldsymbol{T}^{(k)}(u_w, v_h)$ at all of the parameters (u_w, v_h) corresponding to the data points $\boldsymbol{Q}_{wh}$ need to be calculated. This calculation requires a high computational cost when the number of data points is large. Therefore, the T-spline computation strategy has a substantial impact on the efficiency of the algorithm.

As we know, the T-spline (2.1) is point-based, and each T-mesh vertex corresponds to a blending function [13]. In each progressive fitting iteration, the blending functions remain unchanged, and only the T-mesh vertices are changed. Moreover, the numbers of T-mesh vertices and blending functions are much smaller than the number of data points. Therefore, our computation strategy is based on the following data structure *T-node*, which corresponds to a blending function and a T-mesh vertex:

```
struct { double u[1:5];
         double v[1:5];
         double P[1:3];
         vector u_cache;
         vector v_cache} T–node.
```

Here, u and v are the knot vectors along the u- and v-directions; P is the mesh vertex corresponding to the T-node. The two vectors *u_cache* and *v_cache* are explained as follows.

Suppose the blending function corresponding to the current T-node is $B_k(u,v)$, which is the product of two cubic B-spline bases, i.e., $B_k(u,v) = N_k(u)N_k(v)$, where $N_k(u)$ and $N_k(v)$ are defined on the knot vectors

$$[u_{w_0}, u_{w_1}, u_{w_2}, u_{w_3}, u_{w_4}] \quad \text{and} \quad [v_{h_0}, v_{h_1}, v_{h_2}, v_{h_3}, v_{h_4}],$$

respectively. Because of the locality of the B-spline bases, $B_k(u,v) = N_k(u)N_k(v)$ is nonzero over the region $(u_{w_0}, u_{w_4}) \times (v_{h_0}, v_{h_4})$ (refer to Figure 2.1(b) and the mesh vertex $\boldsymbol{P}_8$ therein). Therefore, we store the values $N_k(u_w)$, $w = w_0+1, w_0+2, \ldots, w_4-1$, in the vector *u_cache* and the values $N_k(v_h)$, $h = h_0+1, h_0+2, \ldots, h_4-1$, in the vector *v_cache*.

To calculate the values $\boldsymbol{T}^{(k)}(u_w, v_h)$, $w = 0, 1, \ldots, \varphi-1$, $h = 0, 1, \ldots, \psi-1$, we prepare a $\varphi \times \psi \times 3$ zero array $V[1:\varphi, 1:\psi, 1:3]$. The calculation procedure traverses all T-nodes. For each T-node, the three elements in the T-node structure, i.e., $P[3]$, $N_k(u_w)$ in *u_cache*, and $N_k(v_h)$ in *v_cache*, are multiplied and added to the element $V[w,h,1:3]$. Finally, the array $V[1:\varphi, 1:\psi, 1:3]$ stores the value $\boldsymbol{T}^k(u_w, v_h)$ at (u_w, v_h).

The T-nodes are constructed before each round of progressive fitting iterations. In the iteration procedure, only the T-mesh vertices $P[1:3]$ are changed, whereas the other elements in the T-node remain unchanged.

It should be noted that when the number of T-mesh vertices increases, the nonzero area of each blending function will decrease. Thus, the proposed T-spline computation strategy makes the total amount of computation unchanged, and then the iteration speed of progressive fitting algorithm is steady and insensitive to the increase in the number of T-mesh vertices.

Parallelization. Because the computation of $\boldsymbol{T}^{(k)}(u_w, v_h)$ at (u_w, v_h), corresponding to the data point $\boldsymbol{Q}_{wh}$, is independent of other data points, the computations can easily be run in parallel. In our implementation, we employ 8-thread parallel computing to calculate $\boldsymbol{T}^{(k)}(u_w, v_h)$ using a four-core CPU.

2.6. Storage of T-mesh. As is well known, the control net of a B-spline patch is a structured data point grid. Therefore, to store the B-spline patch, we need only save the data point grid and two knot vectors. However, the T-mesh loses the grid structure, and then we should store the adjacency information among T-nodes, beside the coordinates of T-mesh vertices.

In fact, we employ the following data structure to save the inner T-nodes, which have four adjacent T-nodes:

```
struct { double P[1:3];
         double uv_para[1:2];
         int adj_nodes[1:4]} inner–T–node,
```

where P are the coordinates of the T-node, *uv_para* are the (u, v) parameter values corresponding to the T-node, and *adj_nodes* stores the serials of the four T-nodes adjacent to the current T-node, i.e., its left, right, upper, and lower neighbors.

As to a terminal T-node, such as the node $\boldsymbol{P}_1$ in Figure 1.1(a), which has no left neighbor, we set the element corresponding to its left neighbor in *adj_nodes* as -1 and add a double type array to store the two parameters determined by the ray along the reverse u-direction, i.e., $[u_0, u_1]$.

3. Results and discussion. We have implemented our progressive T-spline data fitting algorithm and run it on a PC with Intel Core2 Quad CPU Q9400 2.66GHz and 4G memory. To validate the efficiency and effectiveness of our method, in this section, we test our method in image fitting. The test results are illustrated in Figures 3.1–3.5. Specifically, our method is employed to fit an image with resolution $\varphi \times \psi$ (i.e., with $\varphi \times \psi$ pixels), where the *rgb*-color values (r_{wh}, g_{wh}, b_{wh}) are taken as the coordinates (x, y, z) of the data points, and their parameters (u_w, v_h) are set just as their indices (w, h), i.e., $(u_w, v_h) = (w, h)$. In all of the results, if not pointed out explicitly, the insertion ratio $\alpha\%$ is taken as 10%.

Comparison with Gauss–Seidel, conjugate gradient, and preconditioned conjugate gradient iteration methods. We compare our progressive fitting algorithm with the Gauss–Seidel, conjugate gradient (CG), and Gauss–Seidel preconditioned CG iteration methods in the examples demonstrated in Figures 3.1, 3.2, and 3.3, whose resolutions are 11747×5400, 3674×2449, and 1680×1050 (refer to Table 3.2), respectively. All of the methods are implemented in C++. While the Gauss–Seidel and Gauss–Seidel preconditioned CG methods are matrix-based, the CG method and our method are both matrix-free. It should be pointed out that, in fitting the image shown in Figures 3.1(left), the first rounds of Gauss–Seidel, CG, and preconditioned CG iterations are not completed in 24 hours.

Figure 3.4 shows the diagrams of the logarithm (base 10) of the average iteration time T for each step in each round of iterations vs. the number of unknown T-mesh vertices for our progressive fitting algorithm, Gauss–Seidel, CG, and preconditioned CG, respectively. While the average iteration time for each step of our progressive fitting is steady and insensitive to the number of T-mesh vertices, ranging from $[4.81, 5.8]$ seconds in Figure 3.4(a) to $[0.94, 1.14]$ seconds in Figure 3.4(b), respectively, the iteration speeds of the Gauss–Seidel, CG, and preconditioned CG methods are made very slow with the increasing number of unknown T-mesh vertices.

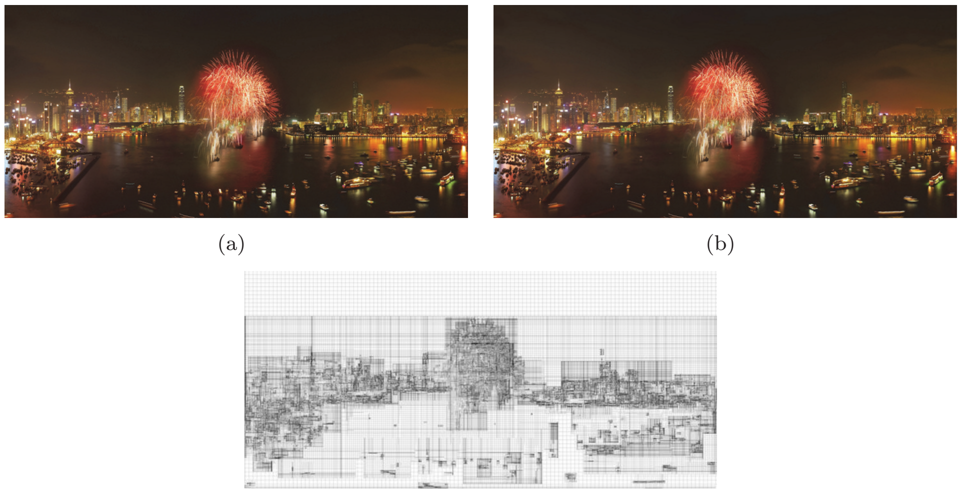


FIG. 3.1. *Fitting a large photographic image "Night Scene" by our progressive T-spline fitting algorithm.* (a) *The original image with* 11747 × 5400 *pixels.* (b) *The reconstructed image from the T-spline with PSNR* 41.4678 *dB.* (c) *The preimage of the generated T-mesh with* 449764 *vertices.*

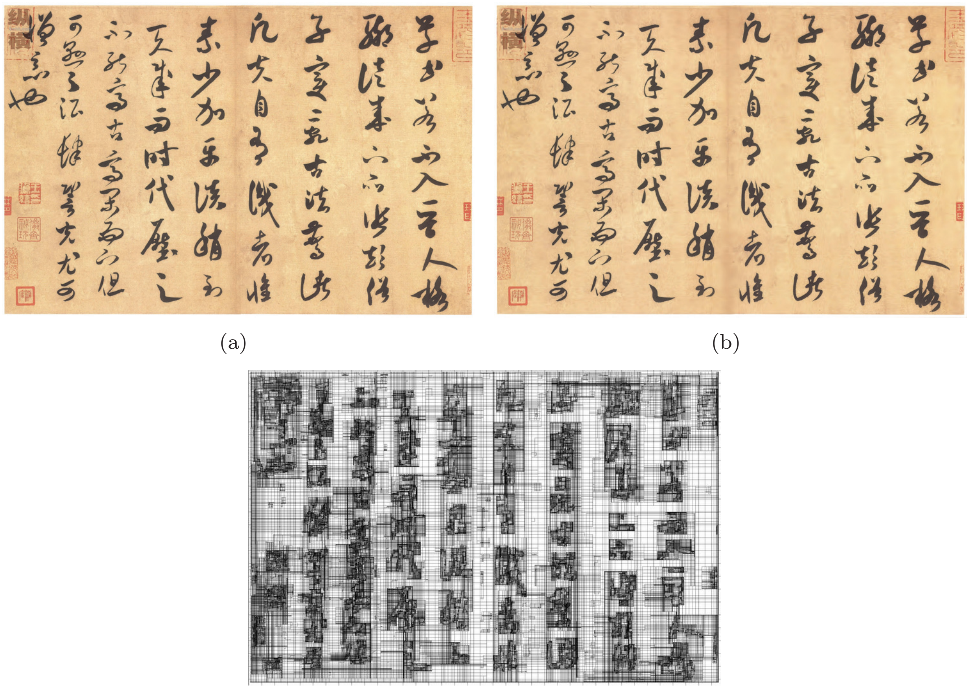
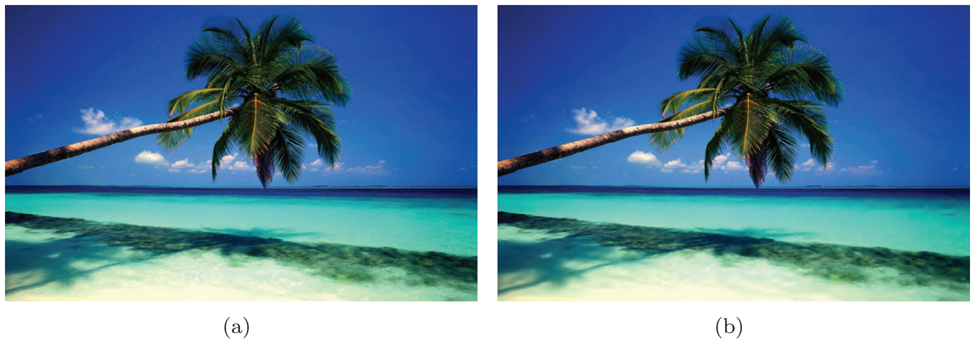


FIG. 3.2. *Fitting a piece of Chinese calligraphy by our progressive T-spline fitting algorithm.* (a) *The original image with resolution* 3674 × 2449. (b) *The reconstructed image from the T-spline.* (c) *The preimage of the generated T-mesh with* 348318 *vertices.*



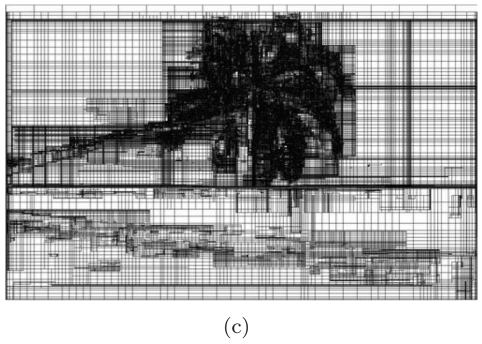


FIG. 3.3. *Fitting a photographic image "Beach" by our progressive T-spline fitting algorithm.* (a) *The original image with resolution* 1680 × 1050. (b) *The reconstructed image from the T-spline.* (c) *The preimage of the generated T-mesh with* 150770 *vertices.*

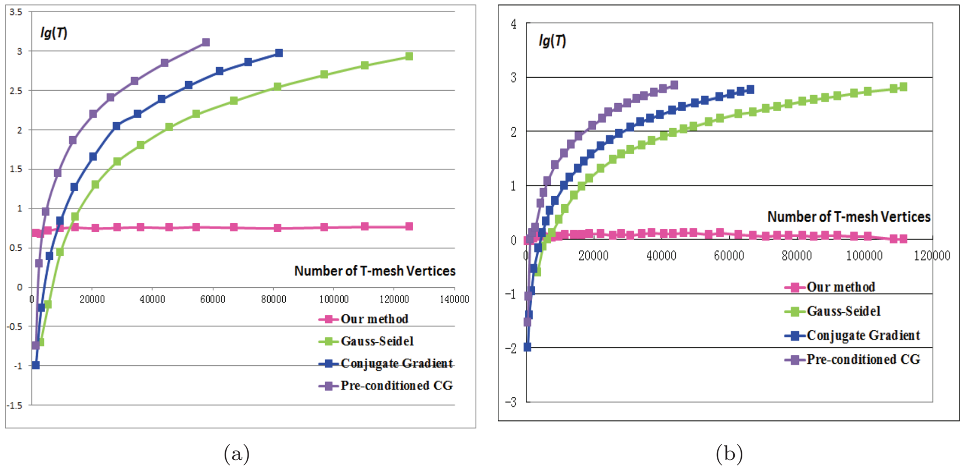


FIG. 3.4. *Diagrams of* lg(*T*) *vs. number of unknown T-mesh vertices, where T is the average iteration time for each step in each round of iterations.* (a) *The diagrams in fitting calligraphy in Figure* 3.2(a). (b) *The diagrams in fitting Beach in Figure* 3.3(a).

Specifically, while the average iteration time cost by the Gauss–Seidel method ranges from $[0.2, 844.667]$ seconds in Figure 3.4(a) to $[0.25, 647.66]$ seconds in Figure 3.4(b), the average iteration time cost by the CG method ranges from $[0.1, 923.544]$ seconds in Figure 3.4(a) to $[0.01, 594.803]$ seconds in Figure 3.4(b), and the time cost

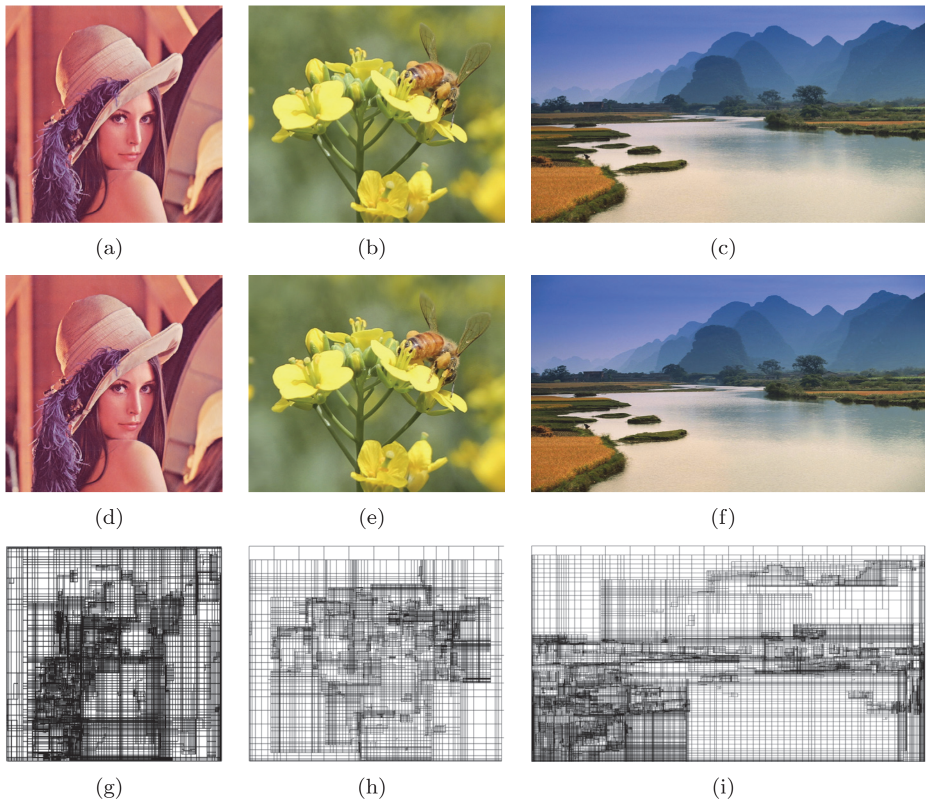


Fig. 3.5. *Fitting photographic images "Lena" (left), "Bee" (middle), and "Landscape" (right) by our progressive T-spline data fitting algorithm.* (a)–(c) *Original images.* (d)–(f) *Reconstructed images from T-splines with insertion ratio* 5%. (g)–(i) *Preimages of T-meshes with insertion ratio* 5%.

by the preconditioned CG method ranges from [0.18, 1126.685] seconds in Figure 3.4(a) to $[0.03, 653.850]$ seconds in Figure 3.4(b). Although the total iteration time of the preconditioned CG method is shorter than that of the CG method to reach the nearly identical fitting precisions (refer to Table 3.1), the average iteration time cost by the preconditioned CG method is longer than that of the CG method (Figure 3.4), because the preconditioned CG method needs extra computation to calculate the Gauss–Seidel preconditioning system of equations, compared with the CG method.

Due to the local support property of T-spline blending functions, when the number of T-mesh vertices increases, the nonzero area of each blending function will decrease. Thus, the T-spline computation strategy proposed in section 2.5 makes the total amount of computation constant, and hence the iteration speed of our progressive fitting algorithm steady and insensitive to the increase in the number of T-mesh vertices. Therefore, our algorithm is more desirable in fitting large-image data with T-splines.

We also notice an interesting phenomenon whereby, when the number of unknown T-mesh vertices is relatively small, the iteration speeds of the CG and preconditioned CG methods are faster than that of the Gauss–Seidel method. However, while the number of unknowns becomes very large, the CG and preconditioned CG methods are slower than the Gauss–Seidel method (see Figure 3.4).

Table 3.1
Time consumed in image fitting with the Gauss–Seidel, CG, and preconditioned CG methods.

	Gauss–Seidel		CG		Preconditioned CG	
	Time[1]	PSNR[2]	Time[1]	PSNR[2]	Time[1]	PSNR[2]
Figure 3.1[3]	N/A	N/A	N/A	N/A	N/A	N/A
Figure 3.2	1492	37.0682	2193	36.5243	1827	37.9528
Figure 3.3	711	41.6087	1327	38.6490	986	39.8506

[1] Time is in minutes.
[2] PSNR (peak signal to noise ratio) is in dB.
[3] The first rounds of iterations are not completed in 24 hours with the Gauss–Seidel, CG, and Gauss–Seidel preconditioned CG methods.

Table 3.2
Statistics for our progressive T-spline fitting algorithm.

	Resolution	#vert.[1]	Time[2]	PSNR[3]
Figure 3.1	11747×5400	449764	318	41.4678
Figure 3.2	3674×2449	348318	81.38	37.5145
Figure 3.3	1680×1050	150770	30.1	42.5845
Figure 3.5 Lena	512×512	41394	3.31	44.8396
Figure 3.5 Bee	1026×789	44144	13	49.6747
Figure 3.5 Landscape	1600×1200	180211	46	46.5826

[1] Number of T-mesh vertices.
[2] Time is in minutes.
[3] PSNR is in dB.

Moreover, Table 3.1 presents the time cost in the image fitting with the Gauss–Seidel, CG, and preconditioned CG methods, respectively. Table 3.2 lists statistics of our progressive T-spline data fitting algorithm. From the runtime listed in Tables 3.2 and 3.1, we can see that the T-spline fitting with our progressive algorithm is faster than that with the Gauss–Seidel, CG, and preconditioned CG methods over 18 times.

Comparison with B-spline representation. As stated above, while the B-spline representation requires lots of superfluous control points just to satisfy its topological constraints, the T-spline can substantially reduce the number of superfluous control points, making the fitting representation much more compact. Thus, the T-spline is more desirable than the B-spline in fitting large data sets. In this section, we compare the T-spline representation with the B-spline representation by the examples illustrated in Figure 3.5 and list the comparison data in Table 3.3. From Table 3.3, we can see that the number of final T-mesh vertices is influenced by the insertion ratio. When the insertion ratio is taken as 5%, the compression ratios of the T-mesh vertex number over the B-spline control point number are reduced to 25%, 22%, and 26% in the three examples in Figure 3.5 (refer to Table 3.3).

Moreover, as illustrated by the preimages of T-meshes in Figures 3.1–3.5, the T-spline representation can capture the features in the images adaptively and automatically.

Data sets with holes. As aforementioned, our method presents a unified framework for fitting data sets with or without holes. Figure 3.6(a) shows an image of a jade caving created in the Western Han Dynasty of China, which has several holes. After generating the initial bicubic B-spline patch, the color data on the holes are eliminated, and only the color data on the jade caving are involved in the computation. Our progressive T-spline data fitting algorithm successfully generates the T-spline fitting result (Figures 3.6(b), 3.6(c)), without any special processing.

Table 3.3
Comparison between number of B-spline control points and that of T-mesh vertices.

	Figure 3.5 Lena		Figure 3.5 Bee		Figure 3.5 Landscape	
PSNR[1]	44.84 ± 2		48.66 ± 2		44.98 ± 2	
B-spline[2]	75898		70488		199101	
Insert. ratio[3]	5%	10%	5%	10%	5%	10%
T-mesh[4]	19038	41394	15617	37277	51656	111688
Comp. ratio[5]	25%	54%	22%	52%	26%	56%

[1] PSNR is in dB; 44.84 ± 2 means that the PSNR values of T-spline and B-spline results lie in the range $[44.84-2,\ 44.84+2]$ dB.
[2] Number of B-spline control points.
[3] Insertion ratio.
[4] Number of T-mesh vertices.
[5] Compression ratio of number of T-mesh vertices over that of B-spline control points.

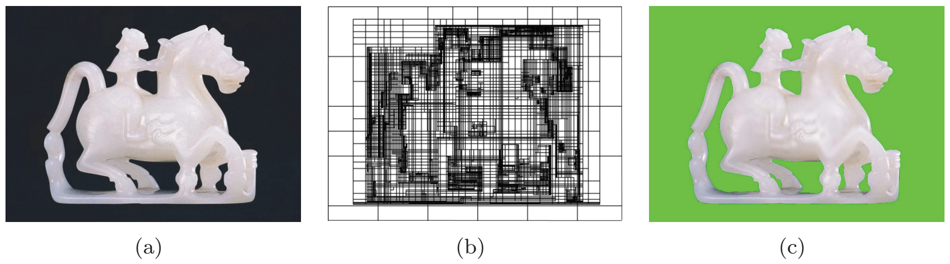


Fig. 3.6. *Fitting image with holes by our progressive T-spline data fitting algorithm.* (a) *Original image with holes.* (b) *Preimage of the T-mesh.* (c) *Reconstructed image from the T-spline.*

4. Conclusion.

In this paper, we develop a progressive T-spline fitting algorithm for fitting large data sets. Starting with an initial bicubic B-spline patch, our algorithm includes two alternately executed procedures: progressive fitting and subregional knot insertion. The progressive fitting algorithm is an iterative method, which is employed to make the current T-spline fit the given data set. If the fitting error does not reach the prescribed precision, the subregional knot-insertion procedure is invoked to insert some knots into the current T-spline, and the progressive fitting algorithm starts again. The fact that the iteration speed of our progressive fitting algorithm is steady and insensitive to the increasing number of unknown T-mesh vertices allows our algorithm to fit large data sets efficiently. In addition, because of the adaptivity of T-spline fitting, this method can significantly reduce the number of control mesh vertices. In this paper, our method is applied in large-image fitting to demonstrate its efficiency and effectiveness. Moreover, because data fitting is a fundamental tool in scientific research and engineering applications, where the fitted data sets are larger and larger, our method will have wide applications.

REFERENCES

[1] A. Amini and F. Marvasti, *Convergence analysis of an iterative method for the reconstruction of multi-band signals from their uniform and periodic nonuniform samples*, Sampling Theory in Signal and Image Processing, 7 (2008), pp. 113–130.
[2] F. Blais, *Review of* 20 *years of range sensor development*, J. Electron. Imag., 13 (2004), pp. 231–240.

[3] A. Buffa, D. Cho, and G. Sangalli, *Linear independence of the t-spline blending functions associated with some particular t-meshes*, Comput. Methods Appl. Mech. Engrg., 199 (2010), pp. 1437–1445.

[4] C. De Boor, *A Practical Guide to Splines*, Appl. Math. Sci. 27, Springer-Verlag, New York, 2001.

[5] P. Dierckx, *Curve and Surface Fitting with Splines*, Oxford University Press, New York, 1995.

[6] Y. He, K. Wang, H. Wang, X. Gu, and H. Qin, *Manifold t-spline*, in Geometric Modeling and Processing - GMP 2006, Lecture Notes in Comput. Sci. 4077, M.-S. Kim and K. Shimada, eds., Springer, Berlin, Heidelberg, 2006, pp. 409–422.

[7] H. Hou and H. Andrews, *Cubic splines for image interpolation and digital filtering*, IEEE Trans. Acoust. Speech Signal Process., 26 (1978), pp. 508–517.

[8] C. Lee, M. Eden, and M. Unser, *High-quality image resizing using oblique projection operators*, IEEE Trans. Image Process., 7 (1998), pp. 679–692.

[9] T.C. Lin, T. Truong, S.H. Chen, L.J. Wang, and T.C. Cheng, *Simplified* 2*-d cubic spline interpolation scheme using direct computation algorithm*, IEEE Trans. Image Process., 19 (2010), pp. 2913–2923.

[10] V. Pereyra and G. Scherer, *Large scale least squares scattered data fitting*, Appl. Numer. Math., 44 (2003), pp. 225–239.

[11] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed., Springer-Verlag, Berlin, 1997.

[12] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[13] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng, and T. Lyche, *T-spline simplification and local refinement*, ACM Trans. Graphics, 23 (2004), pp. 276–283.

[14] T. W. Sederberg, J. Zheng, A. Bakenov, and A. H. Nasri, *T-splines and t-nurccs*, ACM Trans. Graphics, 22 (2003), pp. 477–484.

[15] W. Song and X. Yang, *Free-form deformation with weighted t-spline*, Vis. Comput., 21 (2005), pp. 139–151.

[16] P. Thévenaz and M. Unser, *Separable least-squares decomposition of affine transformations*, in Proceedings of the IEEE International Conference on Image Processing, 1997, pp. 26–29.

[17] M. Unser, *Splines: A perfect fit for signal and image processing*, IEEE Signal Process. Mag., 16 (1999), pp. 22–38.

[18] M. Unser, A. Aldroubi, and M. Eden, *Enlargement or reduction of digital images with minimum loss of information*, IEEE Trans. Image Process., 4 (1995), pp. 247–258.

[19] R.S. Varga, *Matrix Iterative Analysis*, Springer Ser. Comput. Math. 27, Springer, Berlin, 2010.

[20] Y. Wang and J. Zheng, *Control point removal algorithm for t-spline surfaces*, in Geometric Modeling and Processing - GMP 2006, Lecture Notes in Comput. Sci. 4077, M.-S. Kim and K. Shimada, eds., Springer, Berlin, Heidelberg, 2006, pp. 385–396.

[21] J. Zheng, Y. Wang, and H. S. Seah, *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), ACM, New York, 2005, pp. 405–411.