An Image Processing Approach to Feature-Preserving B-Spline Surface Fairing

Taro Kawasaki[a], Pradeep Kumar Jayaraman[b], Kentaro Shida[c], Jianmin Zheng[b], Takashi Maekawa[a,*]

[a]*Department of Mechanical Engineering, Yokohama National University, Japan*
[b]*School of Computer Science and Engineering, Nanyang Technological University, Singapore*
[c]*Armonicos Co., Ltd., Japan*

Abstract

Reverse engineering of 3D industrial objects such as automobiles and electric appliances is typically performed by fitting B-spline surfaces to scanned point cloud data with a fairing term to ensure smoothness, which often smooths out sharp features. This paper proposes a radically different approach to constructing fair B-spline surfaces, which consists of fitting a surface without a fairing term to capture sharp edges, smoothing the normal field of the constructed surface with feature preservation, and reconstructing the B-spline surface from the smoothed normal field. The core of our method is an image processing based feature-preserving normal field fairing technique. This is inspired by the success of many recent research works on the use of normal field for reconstructing mesh models, and makes use of the impressive simplicity and effectiveness of bilateral-like filtering for image denoising. In particular, our approach adaptively partitions the B-spline surface into a set of segments such that each segment has approximately uniform parameterization, generates an image from each segment in the parameter space whose pixel values are the normal vectors of the surface, and then applies a bilateral filter in the parameter domain to fair the normal field. As a result, our approach inherits the advantages of image bilateral filtering techniques and is able to effectively smooth B-spline surfaces with feature preservation as demonstrated by various examples.

Keywords: B-spline fitting, normal field, fairing, feature preservation, image processing, bilateral filtering
2010 MSC: 00-01, 99-00

1. Introduction

Reverse engineering of 3D objects is one of the most workable methods to generate computer-aided design (CAD) models from existing physical objects [1]. It begins with data acquisition process, for example, by laser scanners, to obtain a point cloud representation of a real 3D object, then performs some geometric processes on the point cloud, and finally reconstructs surfaces. Usually, the point cloud is likely to contain outliers and noise due to the acquisition process, or possible artifacts in the physical objects (for instance, in some damaged pieces).

B-splines are often the target representation of the reconstruction for many CAD applications. It is common that the B-spline surface fitting process may induce unwanted "wiggles" in the resulting surface [2], which is particularly exaggerated in the presence of noise. To alleviate this problem, reconstruction techniques usually employ a combination of fidelity and fairness terms to ensure that the surface matches the point cloud as much as possible, and meanwhile keep the surface smooth. A commonly used fairing functional is the thin-plate bending energy and its various variants [3]. However, these functionals are isotropic and do not well preserve sharp features on the underlying surface. While B-spline surface fitting is quite an old topic, reconstructing fair B-spline surfaces with good feature preservation is still nontrivial, and there is little progress on this problem during the last two decades.

By contrast, the research of reconstructing feature preserving 3D meshes is very active in recent years [4, 5, 6, 7]. In particular, filtering the normal

*Corresponding author: Tel.: +81 45 339 3930
Email address: maekawa@ynu.ac.jp (Takashi Maekawa)

field to smooth mesh models seems to have great success [8, 9, 10, 7, 11], since the normal vectors are an important descriptor of the surface shape. Surface normals are an important concept in other computer vision techniques as well, such as shape-from-shading [12, 13, 14, 15].

In this paper, we propose a new approach to constructing fair B-spline surfaces, which is radically different from previous B-spline fitting methods. Our method consists of three steps: The first step is to fit a surface without a fairing term, aiming to capture fine details. The second step, which is the core of our approach, is to smooth the normal field of the constructed surface with feature preservation. We adapt an image processing based bilateral filtering for feature-preserving normal field fairing which works in the parameter domain of the surface. This is inspired by the impressive simplicity and effectiveness of bilateral filtering for image smoothing. The third step is to reconstruct the B-spline surface to match the smoothed normal field. This proposed method is simple in both concept and implementation. The experiments demonstrate the effectiveness, robustness and applicability of the method on a variety of examples. The main contributions of the paper lie in three aspects:

- We propose a three-phase approach for reconstructing fair B-spline surfaces from point cloud. The approach adapts the two techniques: normal field fairing and bilateral filtering, which are widely used in digital geometry processing, to B-spline surface fitting for better preservation of surface details and features.

- We formulate the 3D surface fairing problem as a 2D image smoothing problem, by discretizing the surface in the parameter space, and assigning the normal vector of the surface at the corresponding grid point. Since the parametric speeds [16] of the u and v-isoparametric curves are not constant in general, we propose a strategy to adaptively tessellate the surface and adjust the grid size to ensure that the surface is sampled appropriately.

- We employ the bilateral filtering in the parameter domain to smooth the normal vectors on the 2D grid. Combined with the adaptive tessellation, this approach yields efficient feature preservation.

2. Related work

There is a tremendous amount of literature on surface fitting, and good references for B-spline surface fitting can be found in [17, 18, 19, 1, 20]. This section briefly reviews B-spline surface fairing and feature-preserving smoothing, which are the major focus of our work.

2.1. B-spline Surface Fairing

There are two typical approaches to creating fair B-spline surfaces [21, 22]: (1) fitting surfaces with fairing terms (e.g. [2, 3, 23]) and (2) post-processing (e.g. [24, 21, 25]).

The first approach usually constructs a fairness term and adds it to the objective functional of an optimization problem (see Section 3) to ensure that the fitted surface is free of noise/wiggles. A common formulation of the fairness is the thin plate energy functional which is defined as the surface integral of the sum of two squared principal curvatures of the surface. Due to its highly nonlinear nature, the thin plate energy is often approximated by the integral of the sum of squared second order partial derivatives. For a B-spline surface, this approximate thin plate energy functional is quadratic in the unknown control points, which thus yields a linear system for the optimal solution if the other terms in the objective functional are also quadratic. However, employing the thin plate energy or its approximation will lead to loss of sharp features in the fitting process.

The second approach, to which our method belongs, is to create a surface first and then to smooth it in a later stage. Various post-processing surface fairing methods have been proposed in the literature. For example, an automatic fairing algorithm based on knot removal and knot reinsertion for bicubic B-spline surfaces was introduced in [21]. In general, knot removal will increase the order of smoothness and the knot reinsertion will keep the shape of the surface unchanged. To search for the best knot for removal, a simulated-annealing based search strategy is used, which is a stochastic global optimization method and very slow computationally. Nishiyama et al. [25] proposed another method for removing irregularities of B-spline surfaces via smoothing circular highlight lines. A circular highlight line is formed by the points on a surface, at which an extended surface normal passes a circular light source. The user first interactively identifies

the irregular portions of the pre-images of each circular highlight line and then smooths the portions by cubic Hermite interpolation. Finally the surface is modified by solving a set of nonlinear equations using the Newton-Raphson method to match the smoothed circular highlight lines, which is expected to produce a smoother surface. Wang and Zhang [26] applied a non-uniform spline wavelet transform to simplify and fair NURBS curves and surfaces for modeling, manufacturing, and isogeometric analysis. The wavelet transform decomposes a given function into different frequency components, which could be used to detect and remove high-frequency noise. However, it is not clear whether the proposed method can preserve sharp features.

2.2. Feature Preserving Smoothing

In signal and image processing, denoising/smoothing is a common operation. Many techniques have been developed for this task, among which bilateral filtering is a simple and effective edge preserving method for images [27], which works by combining domain and range filtering. The concept of bilateral filtering has been adapted for mesh denoising [28, 29], in which the information in both tangential and normal directions of the mesh is combined into the filtering kernel. This actually produces a very simple and effective mesh denoising algorithm that preserves the edges well. Inspired by the success of bilateral mesh denoising, various adaptations and extensions have been developed. For example, bilateral filtering has been applied on surface normals of the mesh for feature-preserving mesh denoising [11, 7]. Recently, Zhang et al. [9] proposed a joint bilateral filtering for meshes, where the range filtering is applied on a cleverly constructed guidance normal field, and can provide better feature preserving denoising compared to the traditional case.

Our work is similar to these works in spirit. We smooth the normal field of a B-spline surface and reconstruct the fair B-spline surface from the smoothed normal field. However, different from these works that apply bilateral filtering on the meshes or the surface normals in 3D space, we construct normal map images in 2D parameter domain and employ a bilateral filter in this space to fair the B-spline surface.

We take such a post-processing approach for fairing the B-spline surface instead of, for example, smoothing the point cloud either directly or by triangulation because, even if the input point cloud is taken from the vertices of a smooth triangle mesh, fitting the B-spline surface without a fairness term will still result in wiggles due to the B-spline's tensor product structure and relatively less degrees of freedom of the control points.

3. Overview of the Proposed Surface Fitting

This section presents an overview of our new framework for B-spline surface fitting. The input consists of a set of points, which represents an underlying surface topologically equivalent to a disk, and its corresponding parameterization on a rectangular region. The output is a fair B-spline surface that may contain sharp or semi-sharp features. For surfaces with complicated topology structures, a segmentation process is required and multiple surfaces are needed to fit the data. This paper focuses on single surface fitting for simplicity, however, the method can be extended to handle multiple surface fitting by introducing constraints on the boundaries. Figure 1 illustrates the workflow of the proposed framework. It is composed of three stages: B-spline fitting without a fairness term, normal field fairing and B-spline reconstruction, which are described below.

3.1. B-Spline Fitting without a Fairness Term

We want to find an order (K, L) tensor product B-spline surface to fit the input point cloud $\{\mathbf{Q}_k \mid k = 1, 2, \ldots, P\}$. The B-spline surface is defined by a topologically rectangular set of control points $\mathbf{P}_{ij}$, $0 \leq \mathrm{i} \leq \mathrm{m}$, $0 \leq \mathrm{j} \leq \mathrm{n}$ and two knot vectors $\mathbf{U} = (u_0, u_1, ..., u_{m+K})$ and $\mathbf{V} = (v_0, v_1, ..., v_{n+L})$. The equation of the surface is

$$\mathbf{r}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{ij} N_{i,K}(u) N_{j,L}(v) \ , \qquad (1)$$

where $N_{i,K}(u)$ and $N_{j,L}(v)$ are B-spline basis functions of orders K and L defined over the knot vectors $\mathbf{U}$ and $\mathbf{V}$, respectively.

We adopt the conventional least-squares approximation. The knot vectors are predetermined by uniform knots for simplicity or other non-uniform knot placement approaches [30, 31]. Then, the control points $\mathbf{P}_{ij}$ are found by the following optimization problem:

$$\min \sum_{k=1}^{P} |\mathbf{r}(u_k, v_k) - \mathbf{Q}_k|^2, \qquad (2)$$

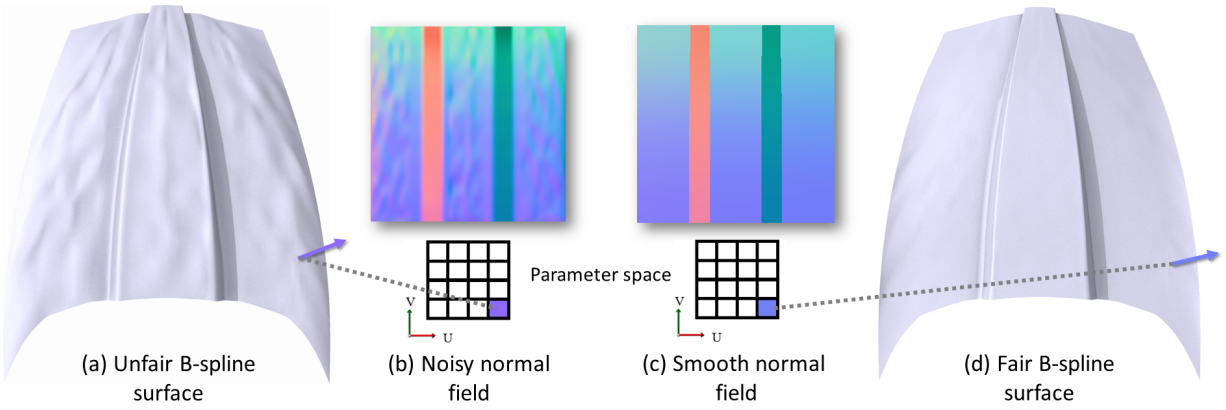


Figure 1: Overview. We begin with an unfair/noisy B-spline surface (HOOD) (a), that is fitted without a fairness term. Next, we discretize its parameter space onto a grid and compute the normal vector at each point, which is visualized as a noisy normal map in (b). We then apply feature preserving smoothing to obtain a smooth normal map (c), and reconstruct a fair B-spline surface (d) that matches these smoothed normals as much as possible.

where (u_k, v_k) are the parameter values of point $\mathbf{Q}_k$. The objective function is quadratic in the control points of the B-spline surface, and the condition for its minimum is the vanishing of the partial derivatives with respect to the control points, which results in a system of linear equations

$$[A_{dist}]\mathbf{d} = \mathbf{b}\ , \tag{3}$$

where A_{dist} is the coefficient matrix, $\mathbf{b}$ is the corresponding right hand side, and $\mathbf{d}$ are the $(n+1) \times (m+1)$ unknown control points. If the deviation criterion is not satisfied, knots are added in both u and v directions where deviations are large, and use for the next surface fit. While conventional surface fitting approaches often add a fairness term, here, we only use the fidelity term to pay more attention to the approximation accuracy, which helps capture the surface detail. However, the obtained surface may contain some unwanted wiggles.

3.2. Feature-Preserving Normal Field Fairing

Observing that when the surface contains wiggles, the normal vectors of the surface will exhibit non-smooth distribution, the process at this stage hence filters the normal field to generate a smooth one. For this purpose, we tessellate the surface regularly in the parameter domain and compute the unit normal vector of the surface at the tessellating points. Then, we view the three coordinates of the normal vectors as 8-bit RGB color values to generate an image for the normal field, and apply feature-preserving filtering based on the bilateral filter to smooth this image. Furthermore, considering possible non-uniform parameterization of the B-spline surface, we propose a strategy to adaptively partition the surface into a set of segments such that each segment has approximately uniform parameterization. With this approach, we obtain multiple images by uniformly tessellating each of these segments, and filter them to produce a set of smoothed images. The technical detail of this stage is elaborated in Section 4.

3.3. Surface Reconstruction from Smoothed Normals

Assume that we have from the previous stage, a set of smoothed images: $\mathbf{I}^h = \{\mathbf{n}^h(k,j) \mid k = 1, \cdots, M^h; j = 1, \cdots, N^h\}$ for $h = 1, \cdots, H$, where H is the number of images, $M^h \times N^h$ is the dimension of image I^h, and $\mathbf{n}^h(k,j)$ are RGB values of I^h actually representing the smoothed normal vector of the surface at the corresponding point with parameter values (u_k^h, v_j^h). Now, we reconstruct the B-spline surface from the smoothed normal field, which is expected to give a fair B-spline surface. The basic idea is to adjust the positions of control points of the B-spline surface so that the new surface normal matches with the smoothed normals at each tessellation point. This is achieved by formulating and minimizing the following energy func-

tional:

$$\sum_{h=1}^{H}\sum_{k=1}^{M^h}\sum_{l=1}^{N^h}\Big(\big(\mathbf{n}^h(k,l)\cdot\mathbf{r}_u(u_k^h,v_l^h)\big)^2+\big(\mathbf{n}^h(k,l)\cdot\mathbf{r}_v(u_k^h,v_l^h)\big)^2\Big)+\sum_{i=0}^{m}\sum_{j=0}^{n}\lambda\Big(P_{ij}-\bar{P}_{ij}\Big)^2, \tag{4}$$

where $\mathbf{r}_u$ and $\mathbf{r}_v$ are the partial derivatives of $\mathbf{r}(u,v)$ with respect to u and v-directions, respectively, and λ is a tradeoff parameter to control how close the new control points P_{ij} are to the old control points $\bar{P}_{ij}$. Here, the first term ensures that the surface matches the normals as much as possible while the second term ensures that the new control points do not deviate too much from the original control points. The objective function is quadratic in the new control points P_{ij} and the necessary conditions for a minimum yield a sparse linear system, which we solve iteratively using the conjugate gradient method.

4. Image Processing based Normal Field Fairing

This section describes the core of the algorithm for normal field fairing, which consists of two components: image generation and bilateral filtering in the parameter domain. First, a set of images are generated whose pixel RGB values represent the unit normal vectors of the surface. Each of these images corresponds to a certain region in the parameter domain of the surface and the union of all these regions covers the whole parameter domain of the surface. Then, bilateral filtering which is adapted to work in the parameter domain is applied to each of the generated images to generate a fair normal field.

4.1. Image Generation

The distribution of the normal vectors reflects the variation of the surface shape, and all the normal vectors of the surface form a normal field. We can discretize the normal field by uniformly tessellating the surface in its parameter domain and computing the unit normal vectors of the surface at the tessellating points. If we treat the coordinates of the normal vectors as RGB values, this results in an image as the normal map. Specifically, to create an $M\times N$ image, we can simply compute the unit normal vector of the surface at the points with parameter values $\left(\frac{i}{M-1},\frac{j}{N-1}\right)$ for $i=0,1,\ldots,M-1$ and $j=0,1,\ldots,N-1$.

This approach is simple but requires that the mapping between the 3D B-spline surface and its 2D parameter domain is roughly uniform. If the parameterization is far from isometric, the uniformly generated image will lose the detail in the areas of high stretch. One brute force approach is to create a high resolution image with very large values of M and N, which requires excessive memory and significantly increases subsequent computational costs.

To address this issue, we propose an adaptive tessellation method that subdivides the parameter domain into different segments such that the parametric speed within each segment is roughly constant. The basic observation is that if the variation of the parametric speed of u and v iso-parametric curves is small, the network formed by the iso-parametric curves on the B-spline surface mapped from a uniform grid in the parameter domain can be considered uniform in the local area of the 3D surface.

Consider a parametric curve $\mathbf{r}(u(t),v(t))$ lying on the B-spline surface $\mathbf{r}(u,v)$ with the parameter domain $\Omega=[u_l,u_r]\times[v_b,v_t]$. Its differential arc length is given by

$$\begin{aligned} ds &= \sqrt{(\mathbf{r}_u\dot{u}+\mathbf{r}_v\dot{v})\cdot(\mathbf{r}_u\dot{u}+\mathbf{r}_v\dot{v})}dt \\ &= \sqrt{Edu^2+2Fdudv+Gdv^2}, \end{aligned} \tag{5}$$

where $E=\mathbf{r}_u\cdot\mathbf{r}_u, F=\mathbf{r}_u\cdot\mathbf{r}_v$ and $G=\mathbf{r}_v\cdot\mathbf{r}_v$ are the coefficients of the first fundamental form. Hence the differential arc lengths of isoparametric curves become $ds=\sqrt{E}du$ along the u-direction and $ds=\sqrt{G}dv$ along the v-direction.

Determining the Image Resolution. We now provide a way to determine M and N, the number of sampling points along the u and v directions in the parameter domain such that the distance between two sampling points on the surface is bounded by a prescribed constant LEN. For an isocurve along the u direction, we have

$$ds=\sqrt{E}du\approx\sqrt{E}\frac{u_r-u_l}{M-1}\le\max_{(u,v)\in\Omega}\sqrt{E}\frac{u_r-u_l}{M-1}.$$

Hence, M can be found as long as it satisfies $\max\limits_{(u,v)\in\Omega}\sqrt{E}\dfrac{u_r-u_l}{M-1}\le LEN$, i.e.,

$$M=\left\lceil\frac{u_r-u_l}{LEN}\max_{(u,v)\in\Omega}\sqrt{\mathbf{r}_u\cdot\mathbf{r}_u}\right\rceil+1, \tag{6}$$

where $\lceil \cdot \rceil$ is the ceil function. Similarly, we can compute N, the number of sampling points along the v direction in the parameter domain, as follows:

$$N = \left\lceil \frac{v_t - v_b}{LEN} \max_{(u,v)\in\Omega} \sqrt{\mathbf{r}_v \cdot \mathbf{r}_v} \right\rceil + 1. \tag{7}$$

With M and N available, it is straightforward to generate uniform samples in the parameter domain along u and v directions, to generate a 2D image, where each pixel is the RGB value of the surface unit normal at the corresponding parameter.

Adaptive Tessellation. As mentioned earlier, directly computing M and N on the entire domain will result in oversampling to compensate for areas of high stretch on the surface. To handle this problem, we adaptively tessellate the domain into multiple images, by checking whether the surface $\mathbf{r}(u, v)$ has approximately constant parametric speeds in the domain Ω. The variation rates of the two parametric speeds are given by:

$$\begin{aligned} \frac{d^2 s}{du^2} &= \frac{d\sqrt{E}}{du} = \frac{\mathbf{r}_{uu} \cdot \mathbf{r}_u}{\sqrt{\mathbf{r}_u \cdot \mathbf{r}_u}}, \\ \frac{d^2 s}{dv^2} &= \frac{d\sqrt{G}}{dv} = \frac{\mathbf{r}_{vv} \cdot \mathbf{r}_v}{\sqrt{\mathbf{r}_v \cdot \mathbf{r}_v}}. \end{aligned} \tag{8}$$

By multivariate calculus, we have

$$\frac{ds(u+\delta, v)}{du} - \frac{ds(u, v)}{du} = \frac{d^2 s(\xi, v)}{du^2}\delta,$$

for some $\xi \in (u, u+\delta)$. If we let $\delta = \frac{u_r - u_l}{M}$, we then have:

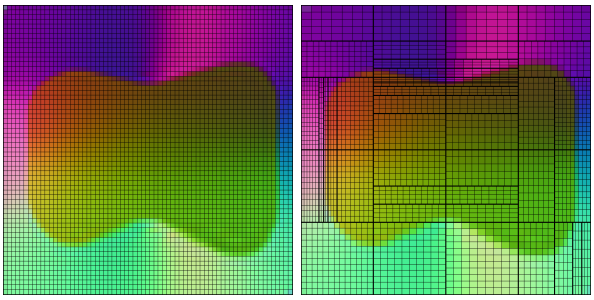
$$\left|\frac{d^2 s(\xi, v)}{du^2}\right| \delta \leq \max_{(u,v)\in\Omega} \left|\frac{d^2 s(u, v)}{du^2}\right| \frac{u_r - u_l}{M}.$$

As long as

$$\frac{1}{M} \max_{(u,v)\in\Omega} \frac{|\mathbf{r}_{uu} \cdot \mathbf{r}_u|}{\sqrt{\mathbf{r}_u \cdot \mathbf{r}_u}} (u_r - u_l) \leq \epsilon, \tag{9}$$

for a prescribed threshold ϵ , the surface $\mathbf{r}(u, v)$ is considered to have approximately constant parametric speed in the u direction in Ω. Otherwise, we can subdivide the interval $[u_l, u_r]$ in the middle and check the inequality in the two sub-intervals. This process can continue till the inequality is satisfied. Similarly, the inequality condition for the v direction is

$$\frac{1}{N} \max_{(u,v)\in\Omega} \frac{|\mathbf{r}_{vv} \cdot \mathbf{r}_v|}{\sqrt{\mathbf{r}_v \cdot \mathbf{r}_v}} (v_t - v_b) \leq \epsilon. \tag{10}$$



(a) Uniform tessellation (b) Adaptive tessellation

Figure 2: Uniform and adaptive tessellation of the parameter domain of the MOUSE model shown with a low-resolution example for clarity. Given a prescribed parameter LEN (here set to 5% of the length of the longest diagonal), adaptive tessellation takes steps that are larger on average (avg. step: 68% of LEN, #pixels: 2622) resulting in fewer number of pixels compared to uniform tessellation (avg. step: 45.6% of LEN, #pixels: 4209).

By combining all the above derivations and discussions, we can easily design a recursive function to subdivide the surface domain and create a set of images, as shown in Algorithm 1 in pseudo-code. An example of such adaptive tessellation of the MOUSE model is shown in Figure 2, and the corresponding result is shown in Section 5. Here, the intersection of black grid lines are the tessellation samples, while each of the tiny rectangles corresponds to a pixel in the normal map image. Note how the total number of pixels is much lesser for the adaptive tessellation even while satisfying the prescribed bound on the step length LEN.

4.2. Bilateral Filtering in Parameter Domain

We now describe our approach to filter the normal map images. Bilateral filtering is a simple and effective edge-preserving image smoothing method [27]. Its main idea is to update the image $I(\mathbf{p})$ at pixel $\mathbf{p} = (i, j)$, as a weighted combination of its neighbors $\mathbf{q} \in N(\mathbf{p})$:

$$\frac{\sum_{\mathbf{q}\in N(p)} W_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) W_{\sigma_c}(\|I(\mathbf{p}) - I(\mathbf{q})\|) I(\mathbf{q})}{\sum_{\mathbf{q}\in N(\mathbf{p})} W_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) W_{\sigma_c}(\|I(\mathbf{p}) - I(\mathbf{q})\|)}, \tag{11}$$

where $W_{\sigma_s}(x) = e^{-\|x\|^2/2\sigma_s^2}$ and $W_{\sigma_c}(c) = e^{-\|c\|^2/2\sigma_c^2}$ are the spatial and color Gaussian kernels with parameters σ_s and σ_c used to penalize large variation in position and intensity, respectively.

Algorithm 1 Pseudo-code for image generation

Require:
LEN: upper bound on step length
ϵ: threshold for parametric speed
$ImageList$: empty list to store images

procedure GenImages($\mathbf{r}$, u_l, u_r, v_b, v_t)
  $\Omega \leftarrow [u_l, u_r] \times [v_b, v_t]$.
  Compute M and N ▷ Eqs.6 and 7.
  $P_u \leftarrow \frac{1}{M} \max\limits_{(u,v)\in\Omega} \frac{|\mathbf{r}_{uu}\cdot\mathbf{r}_u|}{\sqrt{\mathbf{r}_u\cdot\mathbf{r}_u}}$ ▷ Eq.9
  $P_v \leftarrow \frac{1}{N} \max\limits_{(u,v)\in\Omega} \frac{|\mathbf{r}_{vv}\cdot\mathbf{r}_v|}{\sqrt{\mathbf{r}_v\cdot\mathbf{r}_v}}$ ▷ Eq.10
  if $u_r - u_l \le \frac{\epsilon}{P_u}$ and $v_t - v_b \le \frac{\epsilon}{P_v}$ **then**
    Create image I of dimension $M \times N$
    $\hat{n}_{ij} \leftarrow$ unit surface normal at $\mathbf{r}\left(u_l + \frac{i}{M}(u_r - u_l), v_b + \frac{j}{N}(v_t - v_b)\right)$
    $I[i,j] \leftarrow RGB(\hat{n}_{ij})$
    push I into $ImageList$
    return
  else if $u_r - u_l > \frac{\epsilon}{P_u}$ and $v_t - v_b \le \frac{\epsilon}{P_v}$ **then**
    GenImages($\mathbf{r}$, u_l, $\frac{u_l+u_r}{2}$, v_b, v_t)
    GenImages($\mathbf{r}$, $\frac{u_l+u_r}{2}$, u_r, v_b, v_t)
  else if $u_r - u_l \le \frac{\epsilon}{P_u}$ and $v_t - v_b > \frac{\epsilon}{P_v}$ **then**
    GenImages($\mathbf{r}$, u_l, u_r, v_b, $\frac{v_b+v_t}{2}$)
    GenImages($\mathbf{r}$, u_l, u_r, $\frac{v_b+v_t}{2}$, v_t)
  else if $u_r - u_l > \frac{\epsilon}{P_u}$ and $v_t - v_b > \frac{\epsilon}{P_v}$ **then**
    GenImages($\mathbf{r}$, u_l, $\frac{u_l+u_r}{2}$, v_b, $\frac{v_b+v_t}{2}$)
    GenImages($\mathbf{r}$, u_l, $\frac{u_l+u_r}{2}$, $\frac{v_b+v_t}{2}$, v_t)
    GenImages($\mathbf{r}$, $\frac{u_l+u_r}{2}$, u_r, v_b, $\frac{v_b+v_t}{2}$)
    GenImages($\mathbf{r}$, $\frac{u_l+u_r}{2}$, u_r, $\frac{v_b+v_t}{2}$, v_t)
  end if
end procedure

The formulation in Eq. 11 is used in common implementations of the bilateral filter, e.g., OpenCV, and computes the spatial distance between the neighboring pixels in the image space by using the row and column indices. With adaptive tessellation, the images generated by our method are of different resolutions, hence, the distance between neighboring pixels is not constant among different images. In detail, the distance between each of the neighboring pixels is given by the $\frac{1}{M-1}$ along the u-direction and $\frac{1}{N-1}$ along the v-direction, with differing values of M and N for each of the images. Hence, it is not accurate to measure distances using image indices, and we formulate the bilateral filter to work with u and v values in the parameter domain by defining the coordinate of a pixel as $\mathbf{p} = (u_i, v_j)$ in the above formulation. With this modification to the spatial component of the bilateral filter, a fair normal map image can be computed by applying Eq. 11 to each pixel, and then updating the values of all pixels as a group.

Remark. We note that, since it is generally not possible to have an isometric parametrization for 3D shapes with non-zero Gaussian curvature, measuring distances on the parameter domain may seem to be inaccurate. However, since our adaptive tessellation method partitions the parameter domain into sections with roughly constant parametric speeds, we found the distance measurements to be good enough for our problem. To verify this, we replaced the distance component of the above formulation with a spatial Gaussian kernel in 3D space that approximately computes the geodesic distance. By comparing the filtering results, we observed that in terms of fairing the surface, measuring distances on the parameter domain was good enough and did not introduce any unwanted artifacts, thanks to the adaptive tessellation.

Implementation. Recall from Figure 2 that the rectangles bounded by the thick black lines are images with different resolutions. For pixels on the interior of these images, the bilateral filter implementation is similar to conventional image-based filters. For pixels close to and along the boundary, we need to obtain neighbors that lie outside the image. Each image has a fixed step size $1/(M-1)$ and $1/(N-1)$ along the $u-$ and $v-$directions, and its pixels have corresponding (u, v) values. From this, we can easily sample the parameter domain directly

to obtain the pixels that lie outside each image, but are still within the B-spline surface's preimage.

5. Results & Evaluation

In this section, we demonstrate the effectiveness of our algorithm by applying it to five models: TWISTED cuboid, HOOD of an automobile, a computer MOUSE, a FANDISK patch, and a BOAT. We acquired the point cloud of the HOOD model by first 3D printing it, and then using laser scanning as shown in Figure 3. The point clouds of MOUSE, FANDISK, and BOAT are all obtained from mesh models. Table 1 lists the parameters used to generate the results of these models shown in this paper. Here, LEN is given as a normalized value with respect to the surface's longest diagonal, the actual value is given in parentheses; filter size (in pixels) is the neighborhood considered for filtering.

5.1. Computational Time

We implemented our algorithm in C++ on a PC running Windows 10 with a core i7-6800-K 3.40 GHz processor with 32 GB of RAM. Table 2 lists the computational time for fitting with thin-plate energy fairing term and our post-processing surface fairing for the five models. Note that the computational time of our method includes both fitting and the post-processing time. We can observe that the computational time for fitting with thin-plate energy fairing term is almost the same as that of fitting without fairing. While our post-processing surface fairing requires a few extra seconds, it is clear from the results and experiments in Section 5.2 that it significantly preserves features in contrast to the thin-plate functional. Note that we did not intentionally optimize our implementation for best performance, and leave it as future work.

5.2. Visual Inspection

We examine the extent of feature preservation, and the fairness of the resulting surfaces by visualizing the Gaussian curvature color maps and zebra maps (reflection lines). Figure 4 depicts the results of our method applied to the HOOD model. It is apparent from the close up views and zebra maps that our method not only fairs the surface, but also preserves the sharp features of the surface. This is further confirmed by the narrow bands of high Gaussian curvature close to the sharp features. Similarly, we next demonstrate our results for the FANDISK, TWISTED cuboid, computer MOUSE, and BOAT models in Figures 5&6. Again, we can observe that the sharp features are well preserved.

We note that sharp features may at times not lie on the isoparametric lines, e.g., see the MOUSE and BOAT models. Our method works reasonably well even in such cases as shown in Figure 6, however, our optimization in Section 3.3 might possibly result in control points that oscillate around the features. This is a long standing problem in spline fitting, and employing feature-aware surface parametrization techniques for B-splines [32, 33] or T-splines [34] can alleviate the issue.

5.3. Evaluation: Adaptive Tessellation

We now evaluate the effect of our adaptive tessellation method. Recall that the parameter LEN is the upper bound of the step length taken on the surface during tessellation, i.e., the corresponding dimension of each pixel on the surface. To demonstrate the efficiency of our method, we perform both uniform and adaptive tessellation on each of the models, and measure some statistics: number of pixels generated,the average step length of the tessellation, and the computational time for post-processing (does not include the fitting time).

Ideally, we prefer a tessellation that respects the bound set by LEN, and is on average close to this value, while keeping the number of samples as low as possible. From the tessellation statistics shown in Table 3, we can see that images generated using our adaptive tessellation are superior in both these criteria, which corresponds to higher efficiency during filtering and reconstruction. Figure 7 shows that the resulting surfaces are virtually indistinguishable.

5.4. Evaluation: Fitting Error

In TWISTED cuboid, HOOD, MOUSE and FANDISK, the termination criterion for fitting was set such that the maximum and the root mean squared distance errors normalized by the diagonal of the bounding box of each model, are within $\varepsilon_{max} < 0.5$ and $\varepsilon_{rms} < 0.2$, respectively.

We quantitatively evaluate our results with those obtained using the thin-plate fairing term in terms of ε_{max} and ε_{rms}, by computing the corresponding distances from the point cloud, see Table 4. Since our fairing method is based on post-processing, the maximum error of certain models, e.g., HOOD, MOUSE, and TWISTED cuboid, exceeded the termination criterion after the fairing. Therefore, to

Figure 3: Automobile HOOD model. Left-to-right: Ground truth B-spline surface model, created by a 3D printer, and scanned by a laser scanner.

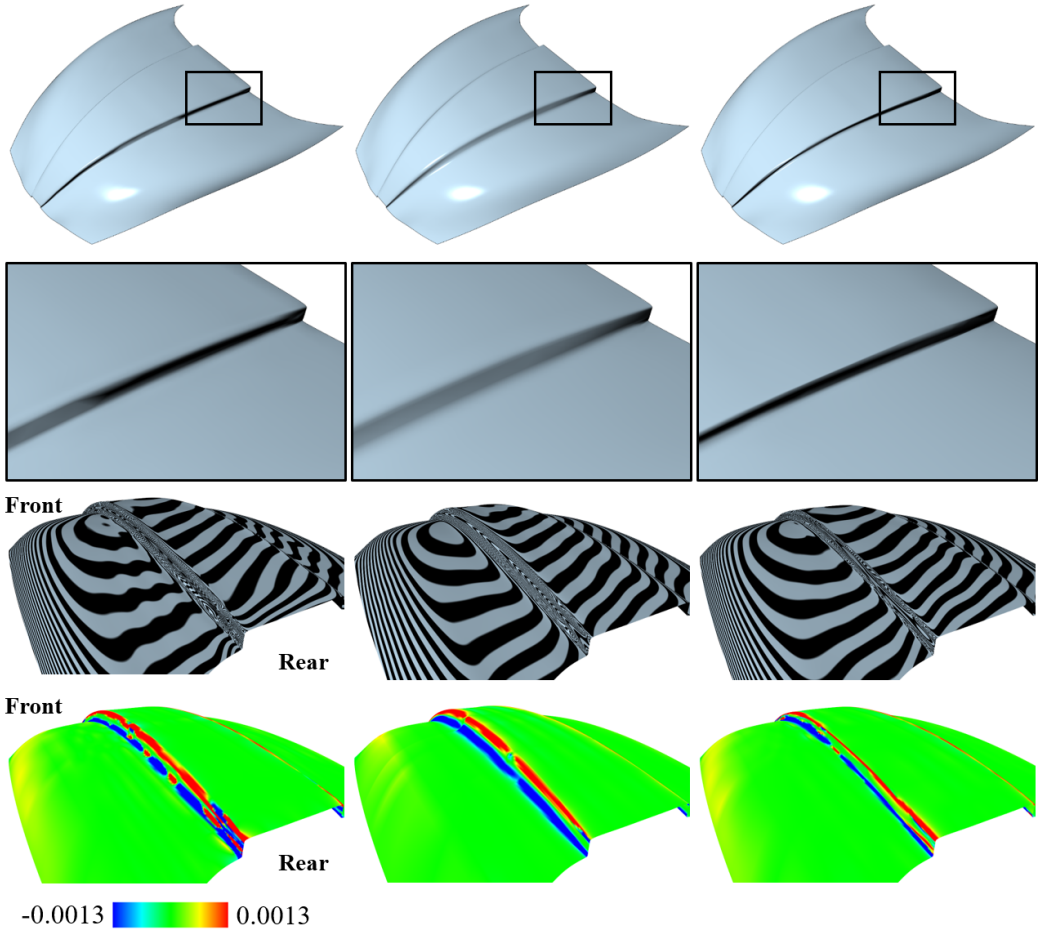


Figure 4: Comparison results for automobile HOOD model. Left-to-right: B-spline surface fitted without a fairing term, fitted with thin-plate energy fairing term, and without a fairing term but post-processed by our method. Top-to-bottom: Shaded images, close-up views, zebra maps and Gaussian curvature color maps.

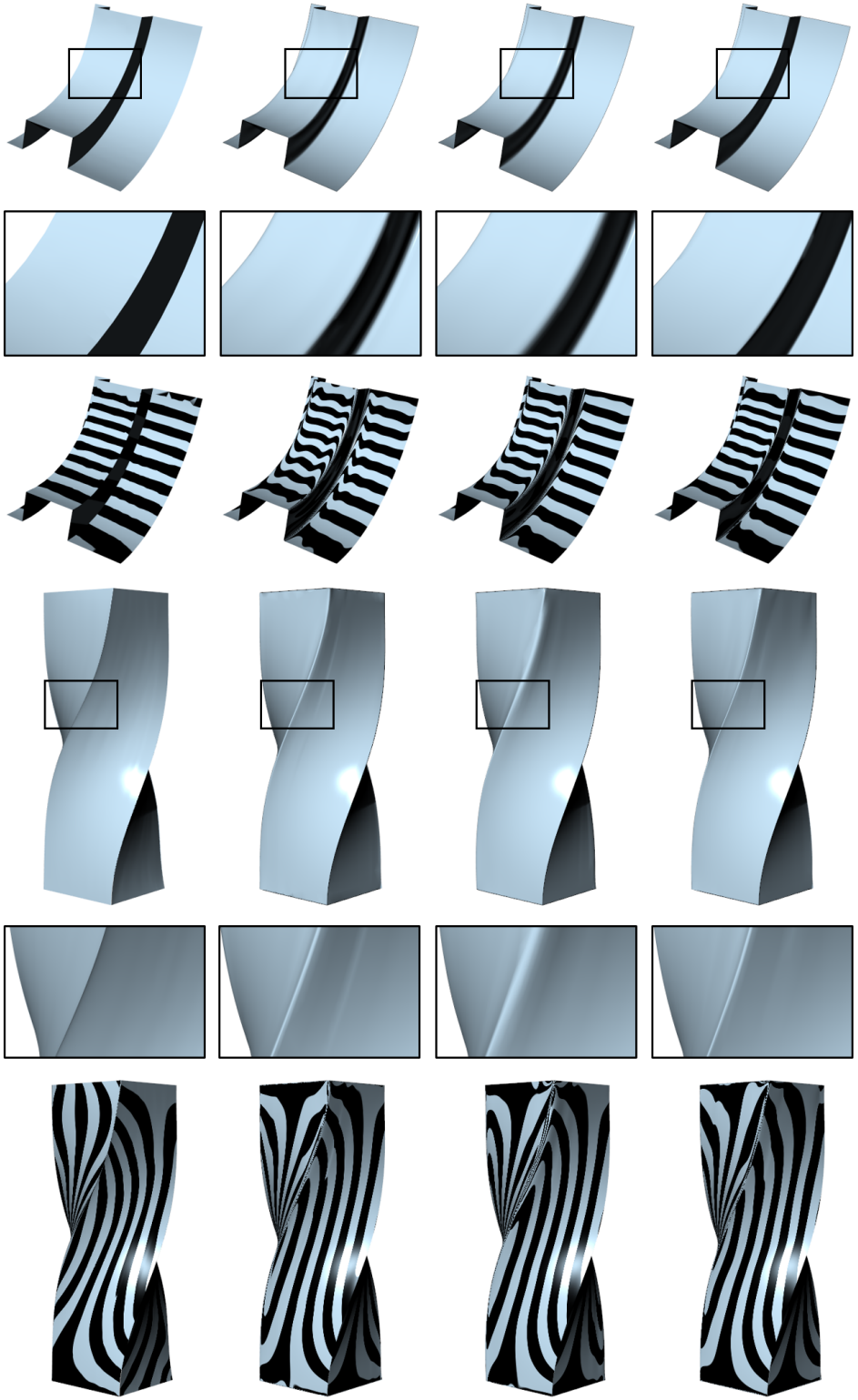


Figure 5: Comparison results for FANDISK and TWISTED cuboid. Left-to-right: Ground truth triangle mesh model, B-spline surface fitted without a fairing term, fitted with thin-plate energy fairing term, and fitted without a fairing term but post-processed by our method. Top-to-bottom: Alternate rows represent rendered results of the models, their corresponding close-up views and zebra maps.

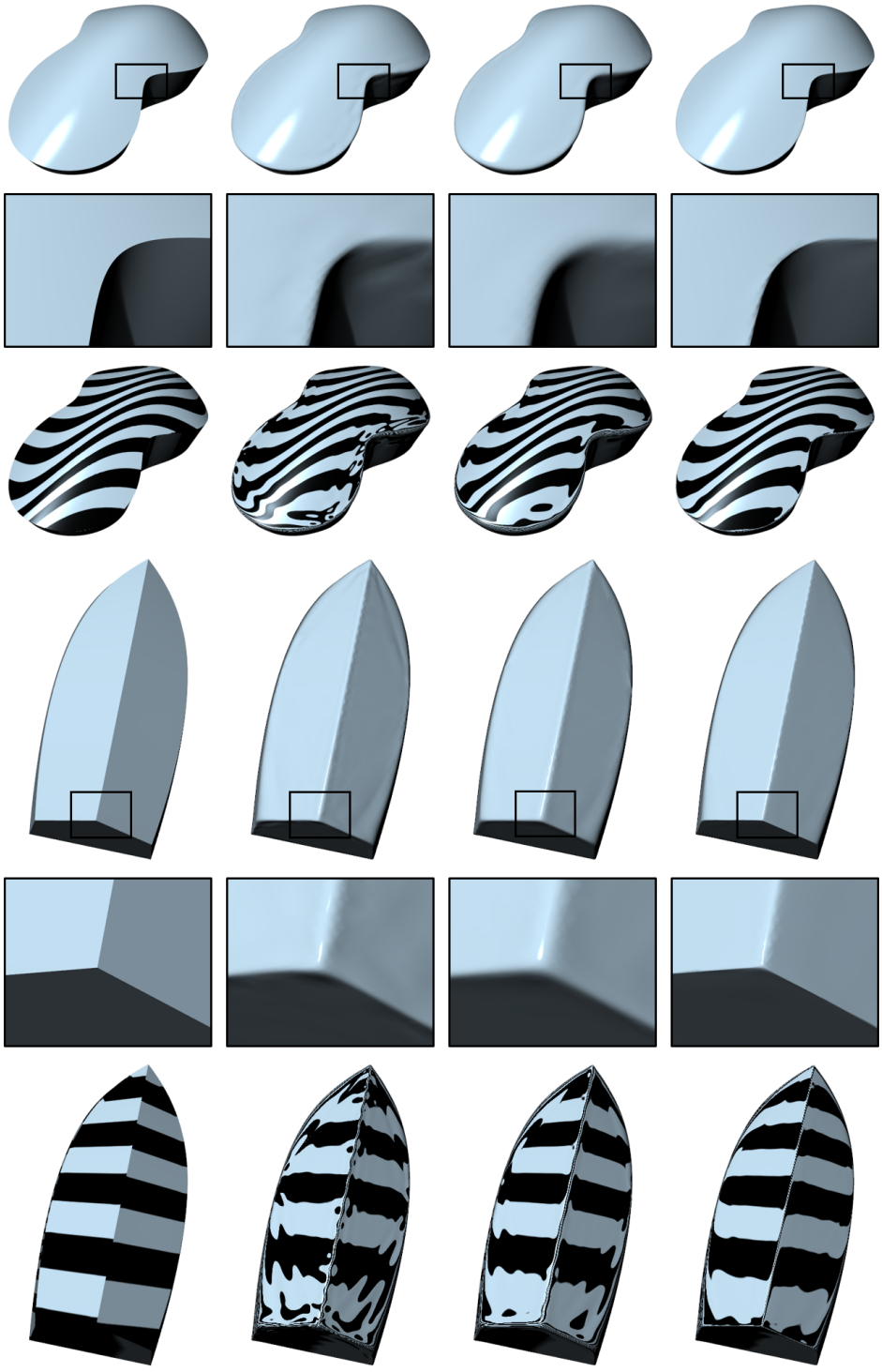


Figure 6: Comparison results for MOUSE and BOAT. Left-to-right: Ground truth triangle mesh model, B-spline surface fitted without a fairing term, fitted with thin-plate energy fairing term, and fitted without a fairing term but post-processed by our method. Top-to-bottom: Rendered results of the models, their corresponding close-up views and zebra maps.

Table 1: Parameters used for generating our results.

Model	Image generation		Filtering				Reconstruction
	LEN	ϵ	#iterations	Filter size	σ_c	σ_s	λ
TWISTED	0.020(0.768)	120	1	6	20.0	0.10	220.0
HOOD	0.030(3.092)	65	2	7	40.0	0.25	150.0
MOUSE	0.015(1.250)	52	1	10	30.0	0.20	100.0
FANDISK	0.020(0.747)	17	1	8	50.0	0.27	100.0
BOAT	0.015(0.730)	11.5	1	10	60.0	0.30	100.0

Table 2: Comparison of computational time.

Model	Running time (s)	
	Thin-plate energy	Our method
TWISTED	57.6	52.9
HOOD	9.46	12.2
MOUSE	60.5	64.1
FANDISK	2.14	3.67
BOAT	12.9	18.4

Figure 7: Visual comparison of BOAT (top) and MOUSE (bottom) surfaces fitted with adaptive tessellation (left) and uniform tessellation (right).

make the comparisons with the thin-plate fairing method on the same ground, we set ε_{max} smaller than 0.5 in the fitting for those models. While the root mean squared distance errors of our results are larger than the thin-plate faired models, these errors are within 0.2, which is generally a sufficient quality for industrial objects.

6. Conclusion

We have described a novel feature-preserving fairing method for B-spline surfaces, which is conceptually simple and easy to implement. Our main contributions include the novel formulation that converts the 3D B-spline surface fairing problem into a 2D image filtering problem, an adaptive tessellation method that generates a set of normal maps with approximately uniform distribution, and bilateral filtering in the parameter domain that preserves features well. We have applied our method to various freeform surfaces with a variety of smooth and sharp features, and demonstrated the versatility of our algorithm through quantitative and visual experiments. We plan to explore ways to extend our method to handle multiple surfaces, NURBS and T-splines in future. Additionally, we would like to experiment with the application of other image processing filters to the post processing step.

Acknowledgements

This research was partially supported by MOE AcRF Tier 1 Grant of Singapore (RG26/15), MultiplAtform Game Innovation Centre (MAGIC) in Nanyang Technological University. MAGIC is funded by the Interactive Digital Media Programme Office (IDMPO) hosted by the Media Development Authority of Singapore. Also, this research was partially supported by YNU ROUTE Programs and YNU Bilateral Collaborative Research Programs. The authors would like to thank Ryosuke Kikuchi and Taketoshi Suzuki for their assistance.

Table 3: Tessellation statistics.

	Uniform			Adaptive		
Model	#pixels	Avg. step	time (s)	#pixels	Avg. step	time (s)
Twisted	57661	0.218	18.7	6820	0.514	2.94
Hood	45981	0.723	14.5	11275	1.465	4.47
Mouse	32448	0.651	9.64	26225	0.737	8.15
Fandisk	5510	0.481	1.87	4547	0.539	1.71
Boat	34780	0.401	10.8	23147	0.530	8.61

Table 4: Comparison of distance errors.

	Thin plate energy			Our method		
Model	#CP	Max error (%)	RMSE (%)	#CP	Max error (%)	RMSE (%)
twisted	47×14	0.488	0.0444	45×12	0.499	0.118
Hood	23×32	0.499	0.108	21×30	0.436	0.121
Mouse	30×36	0.494	0.0578	30×36	0.448	0.153
Fandisk	12×28	0.455	0.0882	11×27	0.455	0.0609
Boat	34×34	0.491	0.0407	32×32	0.467	0.109

References

[1] T. Várady, R. Martin, J. Cox, Reverse engineering of geometric models–an introduction, Comput.-Aided Design 29 (4) (1997) 255–268.

[2] M. Eck, H. Hoppe, Automatic reconstruction of B-spline surfaces of arbitrary topological type, in: Proc. SIGGRAPH, Annual Conference Series, ACM, 1996, pp. 325–334.

[3] G. Greiner, Variational design and fairing of spline surfaces, Comput. Graph. Forum 13 (3) (1994) 143–154.

[4] X. Lu, X. Liu, Z. Deng, W. Chen, An efficient approach for feature-preserving mesh denoising, Optics and Lasers in Engineering 90 (2017) 186–195.

[5] S. K. Yadav, U. Reitebuch, K. Polthier, Mesh denoising based on normal voting tensor and binary optimization, IEEE Trans. Vis. Comput. Graph. PP (99) (2017) 1–1.

[6] M. Wei, J. Yu, W.-M. Pang, J. Wang, J. Qin, L. Liu, P.-A. Heng, Bi-normal filtering for mesh denoising, IEEE Trans. Vis. Comput. Graph. 21 (1) (2015) 43–55.

[7] Y. Zheng, H. Fu, O. Au, C.-L. Tai, Bilateral normal filtering for mesh denoising, IEEE Trans. Vis. Comput. Graph. 17 (10) (2011) 1521–1530.

[8] D. Coeurjolly, M. Foare, P. Gueth, J. Lachaud, Piecewise smooth reconstruction of normal vector field on digital data, in: Comput. Graph. Forum (Pac. Graph.), Vol. 35, Wiley Online Library, 2016, pp. 157–167.

[9] W. Zhang, B. Deng, J. Zhang, S. Bouaziz, L. Liu, Guided mesh normal filtering, Comput. Graph. Forum 34 (7) (2015) 23–34.

[10] L. Zhu, M. Wei, J. Yu, W. Wang, J. Qin, P.-A. Heng, Coarse-to-fine normal filtering for feature-preserving mesh denoising based on isotropic subneighborhoods., Comput. Graph. Forum 32 (7) (2013) 371–380.

[11] K. Lee, W. Wang, Feature-preserving mesh denoising via bilateral normal filtering, in: Ninth Intl. Conf. Comput. Aided Design and Comput. Graph., IEEE, 2005.

[12] C. Esteban, G. Vogiatzis, R. Cipolla, Multiview photometric stereo, IEEE Trans. Pattern Anal. Mach. Intell. 30 (3) (2008) 548–554.

[13] R. Zhang, P.-S. Tsai, J. Cryer, M. Shah, Shape-from-shading: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 21 (8) (1999) 690–706.

[14] R.Woodham, Photometric method for determining surface orientation from multiple images, in: B. K. P. Horn, M. J. Brooks (Eds.), Shape from Shading, MIT Press, 1989, pp. 513–531.

[15] S. R. Richter, S. Roth, Discriminative shape from shading in uncalibrated illumination, in: Proc. IEEE Conf. Comput. Vis. Pattern Rec., 2015, pp. 1128–1136.

[16] N. M. Patrikalakis, T. Maekawa, Shape Interrogation for Computer Aided Design and Manufacturing, Springer-Verlag, Heidelberg, 2002.

[17] V. Weiss, L. Andor, G. Renner, T. Várady, Advanced surface fitting techniques, Comput. Aided Geom. Design 19 (1) (2002) 19–42.

[18] J. Hoschek, D. Lasser, Fundamentals of Computer Aided Geometric Design, A. K. Peters, Wellesley, MA, 1993.

[19] L. Piegl, W. Tiller, The NURBS Book, Springer, New York, 1995.

[20] Y. Kineri, M. Wang, H. Lin, T. Maekawa., B-spline surface fitting by iterative geometric interpolation/approximation algorithms, Comput.-Aided Design 44 (7) (2012) 697–708.

[21] S. Hahmann, S. Konz, Fairing bi-cubic b-spline surfaces using simulated annealing, Curves and Surfaces with Applications in CAGD (1997) 159–168.

[22] E. Sariöz, An optimization approach for fairing of ship hull forms, Ocean Engineering 33 (16) (2006) 2105–2118.

[23] U. Dietz, Fair surface reconstruction from point clouds, in: Proc. Intl. Conf. Mathematical Methods for Curves and Surfaces II, 1998, pp. 79–86.

[24] J. Hadenfeld, Local energy fairing of b-spline surfaces, Mathematical Methods for Curves and Surfaces (1995) 203–212.

[25] Y. Nishiyama, Y. Nishimura, T. Sasaki, T. Maekawa, Surface faring using circular highlight lines, Comput.-Aided Design Appl. 4 (1-4) (2007) 405–414.

[26] W. Wang, Y. Zhang, Wavelets-based NURBS simplification and fairing, Computer Methods in Applied Mechanics and Engineering 199 (5) (2010) 290–300.

[27] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in: Sixth Intl. Conf. Computer Vision 1998, IEEE, 1998, pp. 839–846.

[28] T. Jones, F. Durand, M. Desbrun, Non-iterative, feature-preserving mesh smoothing, ACM Trans. Graph. (SIGGRAPH) 22 (3) (2003) 943–949.

[29] S. Fleishman, I. Drori, D. Cohen-Or, Bilateral mesh

denoising, ACM Trans. Graph. (SIGGRAPH) 22 (3) (2003) 950–953.

[30] H. Kang, F. Chen, Y. Li, J. Deng, Z. Yang, Knot calculation for spline fitting via sparse optimization, Computer-Aided Design 58 (2015) 179–188.

[31] F. Hou, Y. He, H. Qin, A. Hao, Knot optimization for biharmonic B-splines on manifold triangle meshes, IEEE Trans. Vis. Comput. Graph. 23.

[32] Y.-K. Lai, S.-M. Hu, H. Pottmann, Surface fitting based on a feature sensitive parametrization, Comput.-Aided Design 38 (7) (2006) 800–807.

[33] Y. Zhang, J. Cao, Z. Chen, X. Li, X.-M. Zeng, B-spline surface fitting with knot position optimization, Computers and Graphics (SMI) 58 (2016) 73–83.

[34] Y. Wang, J. Zheng, Curvature-guided adaptive T-spline surface fitting, Comput.-Aided Design 45 (8) (2013) 1095–1107.