Multi-Dimensional Dynamic Programming in Ruled Surface Fitting

Charlie C. L. Wang

Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
Corresponding Author; Tel: (852) 3943 8052; Fax: (852) 2603 6002

Gershon Elber

Department of Computer Science, The Technion Israel Institute of Technology, Haifa 32000, Israel

Abstract

Ruled surfaces play an important role in many manufacturing and construction applications. In this work, we explore a multi-dimensional dynamic programming based ruled surface fitting scheme to a given freeform rational surface, S. Considering two initial opposite boundaries of S, sampled into a discrete piecewise linear polyline representation, the ruled surface fitting problem is reduced to a pairing-search between the polylines and elevations above the polylines, in the normal directions of S. *A four-dimensional dynamic programming* solution is sought for the four dimensions prescribed by the two polylines and the two elevation levels along the surface normals. This multi-dimensional dynamic programming is evaluated using highly parallel algorithms running on GPUs that ensures the best fit to the sampled data. In order to evaluate the fitting error with respect to S, we derive a scheme to compute a bound from above on the maximal error between a bilinear surface patch (formed by two consecutive point-pairs) and its corresponding surface region on S. Surface-surface composition is employed to extract the corresponding surface region on S to compare against. Finally, the above ruled surface fitting approach is also extended into a discrete algorithm to find the non-isoparametric subdivision curve on S when a discrete recursive piecewise-ruled surface fitting is considered. A five- or seven-dimensional dynamic programming solution is employed toward this end and once again, surface-surface composition is employed to extract the two subdivided patches as tensor products.

Key words: Ruled surface fitting, Multi-dimensional dynamic programming, Surface-surface composition, GPU algorithms

1. Introduction

Ruled surfaces are widely used in many applications in manufacturing as well as civil engineering application. In manufacturing, while multi-axis CNC processes are well known, other manufacturing technologies, such as wire *Electrically Discharged Machining* (EDM) [1] and 5-axis CNC flanking milling [2–4] are also in extensive use. In wire EDM, a conducting wire is discharging electricity against a conductive stock and evaporates and cuts ruling lines in the material. Tool side CNC machining (e.g., flank milling) exploits the linear edge of the rotating tool to similarly cut along a line after carefully considering the non-zero thickness of the tool. Interestingly enough, other manufacturing processes also cut along lines in 3D. Examples include CNC water jet cutting as well as laser cutting. In building construction, ruled surfaces are used for the realization of free-form architecture [5]. Recently, the use of hot wire cutting in styrofoam was also proposed towards mold making of surfaces tiles in architecture design [6].

Within all above applications, it is interesting that the following *Ruled Surface Fitting* (RSF) is still a highly investigated problem:

Problem (RSF): Given a parametric surface $\mathbf{S}(u, v)$, find the *best* ruled surface fit to $\mathbf{S}$ under some metric.

A good solution to the above RSF problem would clearly increase the usability and quality of all aforementioned technologies for fabricating a given surface $\mathbf{S}$ by swept cutting lines. The accuracy of the resulting artifact would greatly benefit from such an optimal ruling fit. However, as a ruled surface is hyperbolic in general [7], one can expect that only a hyperbolic patch (or parabolic) $\mathbf{S}$ will benefit from such a ruled surface fitting. Forced to use a line-cutting based technology that constructs ruled surfaces, the benefits for elliptical surface regions cannot be significant. Nevertheless, one can still hope to find the optimal ruled surface fit to any surface $\mathbf{S}$ under this constrained manufacturing technologies. If multiple ruled surface patches may be tiled together, while fitting $\mathbf{S}$, the solution of the best *Ruled Surface Partitioning* (RSP) problem is also highly desirable:

Problem (RSP): Given a parametric surface $\mathbf{S}$, find the non-isoparametric partitioning of $\mathbf{S}$ into two sub-patches $\mathbf{S}_L$ and $\mathbf{S}_R$ (or more generally several sub-patches), that is best fitted by two (several) piecewise ruled surfaces, minimizing some metric in RSF.

Email addresses: cwang@mae.cuhk.edu.hk (Charlie C. L. Wang), gershon@cs.technion.ac.il (Gershon Elber).

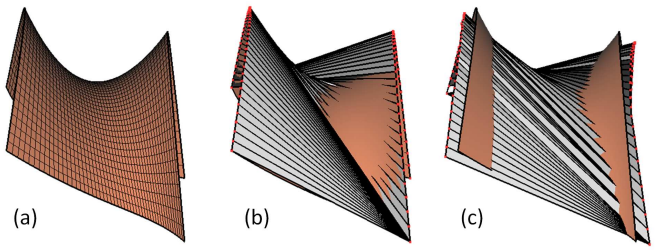


Fig. 1. An example of (multi-dimensional) dynamic programming (DP) based ruled surface fitting: (a) the given freeform surface S, (b) the fitted ruled surface interpolating the boundary curves of $\mathbf{S}$ (which is obtained by [13] – a 2D-DP with 50×50 sampled points), and (c) the ruled surface fitting by also elevating sampled points along the normal of $\mathbf{S}$ (which is solved using a 4D-DP with 50^4 samples). The 4D-DP based ruled surface fitting reduces the Hausdorff distance error by 32.8% compared to the 2D-DP based ruled surface fitting.

Finding the global optimum for RSF and RSP is difficult; unlike the prior work based on numerical optimization, this paper presents a ruled surface fitting scheme for solving the RSF and RSP problems in the discrete domain formed by sample points. The scheme finds an optimal ruled surface fitting once the discrete piecewise linear sampled sets of the boundaries of $\mathbf{S}$ and the possible elevation (again discrete) of these points along the local normal of $\mathbf{S}$, are given. Each discrete ruling line is defined through two elevated sampled points on two opposite boundaries. The elevation is along the normal of $\mathbf{S}$ and is also discretized.

Two consecutive ruling lines above the discrete sampled point set define a bilinear surface patch $\mathbf{B}$ for which the corresponding patch of $\mathbf{S}$, $\mathbf{S}_b$, is extracted using a surface-(bilinear) surface composition. An upper bound on the Hausdorff distance between $\mathbf{B}$ and $\mathbf{S}_b$ is established by computing a bound on $\|\mathbf{B}(u,v) - \mathbf{S}_b(u,v)\|$.

A four-dimensional dynamic programming problem is defined for all possible pairs of points and all possible allowed elevations. The best ruling fit to this discrete arrangement is then determined by solving a four-dimensional dynamic programming (4D-DP). A simple example can be seen in Fig. 1. The DP algorithm has two phases. First a multi-dimensional table is computed for all possible pairing and elevations, a 4D table in this case. Then, a path tracking is conducted from the source (the first ruling line at the beginning of the two boundaries) to the destination (the last ruling line at the end of the two boundaries). Special treatment must be made for the end conditions as the first and last ruling lines can be in arbitrary elevations. Further, if $\mathbf{S}$ is a periodic surface, the first and the last ruling lines must be the same ruling line.

Both DP steps are of a complexity order that depends on the dimension of the built table. If each side of the table is of length n, the 4D-DP has complexity of $O(n^4)$. However, the computation of each entry in the table requires an upper bound on the Hausdorff distance of $\|\mathbf{B}(u,v) - \mathbf{S}_b(u,v)\|$ and the construction time of the DP table governs the entire computation costs. Therefore, we employ highly parallel algorithms running on *Graphics Processing Units* (GPUs) to speed up the construction of the DP table.

1.1. *Related work*

To approximate a general surface, $\mathbf{S}$, by ruled surface(s) $\mathbf{R}_i$, $\mathbf{S}$ can be divided recursively into strips along isoparametric directions, as in [2], and then each of the strips is fitted with a ruled surface. In [8], a surface subdivision with similar ruled surface fitting goals is made but along isophotes, which bounds the normal deviations but not the maximal deviation between $\mathbf{S}$ and $\mathbf{R}_i$. In both cases, the algorithm is greedy and the RSF/RSP computations are not optimal. Approximation of a general surface using ruled surfaces is also considered in [9] by solving a non-linear optimization function that minimizes distances between the tangent planes of the original surface and the fitted ruled surface over the domain. The special case of a surface of revolution is also considered in [9], exploiting fitted hyperboloid of one sheet that are ruled surfaces. Some work on fitting ruled surfaces was also done by using a curve representation on the dual Gaussian sphere of a surface (ref. [10–12]).

The research of [5, 16] focuses on using continuous non-linear optimization methods to solve the RSF problem over sampled point sets from the input surface $\mathbf{S}$. They conducted a sequence of quadratic programs to solve the optimal surface problem by the *Squared Distance Minimization* (SDM) [17] and applications in architecture has been demonstrated. A given surface is divided into strips in [16] by the asymptotic curves computed from the level-set of field of asymptotic directions – tangents to the intersection curves between the input surface and the tangent plane at a surface point. Overall, the result has not Hausdorff distance guarantee to/from the original input surface. Further, this is different from our method, were we tackle the RSP problem in a discrete domain.

While not much can be found on ruled surface fitting to general surfaces, there is also a large body of research on piecewise developable surface fitting to general surfaces [14, 15, 18–23]. Being a sub-class of ruled surfaces, developable surfaces are also ruled surfaces. In [15, 20], the input general surface is first divided into strips (along isoparametric curves in [15]), which are then fitted with developable surfaces.

Although polygonal meshes are not in the scope of this work, the construction of ruled and developable sheets for a mesh representation has captured much attention. Wang and Tang [13] introduced a fitting of a polygonal strip to a ruled surface and employs the shortest path algorithm to find a best fitting on a graph scored by the fitting error, which is in fact a 2D-DP. The resultant polygonal strips interpolate the boundary curves of the given surface $\mathbf{S}$, ending up in a larger fitting error (see Fig. 1 as an example). Mesh surfaces are also processed in [21, 22] under nonlinear optimization framework into shapes that converge to developable surfaces. In [19], the mesh is divided into polygonal strips and each strip is then laid out flat as a developable sheet. In [18], the mesh model is first decomposed into cylindrical shapes that are then fitting with developable sheets. Other segmentation methods for using developable

Table 1
Summary of Acronyms

Acronyms	Full Description
RSF	Ruled Surface Fitting
RSP	Ruled Surface Partition
DP	Dynamic Programming
GPU	Graphics Processing Units
LSAD	Line Surface Approximate Distance
SSHB	Surface-Surface Hausdorff-distance Bound
VD	Voronoi Diagram

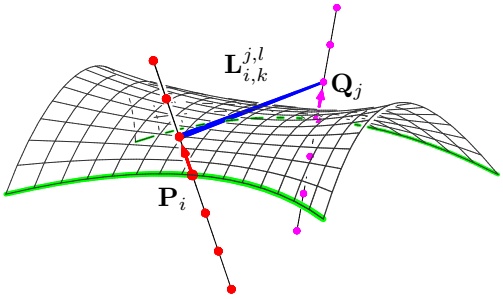


Fig. 2. An illustration for the points elevation: $\mathbf{P}_i \rightarrow \mathbf{P}_{i,k}$ and $\mathbf{Q}_j \rightarrow \mathbf{Q}_{j,l}$.

mesh surface to approximate a given freeform surface can be found in [23, 24].

1.2. *Contributions*

We present algorithms for solving the RSF and the RSP problems in a discrete domain formed by sampling points over freeform input surfaces. The main contributions of our work include:

- An approach is introduced to find the best ruled surface fitting for a discrete, piecewise linear, sampled set of a given general freeform surface, using a four-dimensional dynamic programming scheme.
- A method is presented to provide a guaranteed yet fairly tight upper bound on the Hausdorff distance between the bilinear patch formed out of two adjacent ruling lines and the corresponding region in the input surface $\mathbf{S}$.
- GPU-based algorithms are employed to speed up the evaluation of multi-dimensional DP table.
- The presented ruled surface fitting approach based on dynamic programming is also extended to solve the RSP problem by a 5D-DP to 7D-DP (which allows to elevate boundary curves). A locally-optimal partitioning curve can be determined for a given surface, $\mathbf{S}$, so two ruled surfaces can be fitted to the two subdivided patches, while minimizing some fitting error metric.

The rest of the paper is organized as follows. We present the multi-dimensional DP based method for solving the RSF problem in Section 2, with a guaranteed bounds on the Hausdorff error. The GPU-based algorithms for speeding up the computation are introduced in Section 3. The extension of this multi-dimensional DP technique to compute a solution to the RSP problem is derived in Section 4. Examples and performance evaluations of our approach are described in Section 5. Finally, we conclude in Section 6. Acronyms will be used in the rest of this paper – a summary is given in Table 1.

2. Error-bounded Ruled Surface Fitting

In order to offer a viable solution to the RSF problem, we discretize the solution domain. Without loss of generality, the $v_{\min}$ and $v_{\max}$ boundary curves of a tensor product parametric surface $\mathbf{S}(u, v)$ are denoted by $\mathbf{C}_0(u) = \mathbf{S}(u, v_{\min})$ and $\mathbf{C}_1(u) = \mathbf{S}(u, v_{\max})$ with $u \in [0, 1]$. Let $\mathbf{C}_0(u_i)$ and $\mathbf{C}_1(u_j)$ $(i, j \in [0, n-1])$ be n uniformly sampled points along $\mathbf{C}_0$ and $\mathbf{C}_1$:

$$\begin{aligned} \mathbf{P}_i &= \mathbf{C}_0(u_i) = \mathbf{C}_0((1 - \tfrac{i}{n-1})u_{\min} + \tfrac{i}{n-1}u_{\max}), \\ \mathbf{Q}_j &= \mathbf{C}_1(u_j) = \mathbf{C}_1((1 - \tfrac{j}{n-1})u_{\min} + \tfrac{j}{n-1}u_{\max}). \end{aligned} \tag{1}$$

Further, let $\hat{\mathbf{n}}(u_0, v_0)$ denote the unit normal vector of $\mathbf{S}(u, v)$ at (u_0, v_0). The surface normal space above or below $\mathbf{C}_0(u)$ is discretized into $n \times (2m + 1)$ sample points, $\mathbf{P}_{i,k}$, with $i \in [0, n-1]$ marches along $\mathbf{C}_0$ and $k \in [-m, m]$ along the surface normals, as

$$\mathbf{P}_{i,k} = \mathbf{C}_0(u_i) + k\tau\hat{\mathbf{n}}(u_i, v_{\min}), \tag{2}$$

where τ governs the step size along the normal direction. These points are called P-points.

In a similar fashion, we define a set of Q-points around $\mathbf{C}_1(u)$ as

$$\mathbf{Q}_{j,l} = \mathbf{C}_1(u_j) + l\tau\hat{\mathbf{n}}(u_j, v_{\max}), \tag{3}$$

where $j \in [0, n-1]$ and $l \in [-m, m]$. An illustration for this point elevation is given in Fig.2.

Every $(\mathbf{P}_{i,k}, \mathbf{Q}_{j,l})$ pair defines a ruling line, $\mathbf{L}_{i,k}^{j,l} = \mathbf{P}_{i,k}\mathbf{Q}_{j,l}$, with $O(n^2m^2)$ possible ruling lines in all. Once discretized, the RSF problem can be reduced to finding a sequence of ruling line segments Γ:

$$\Gamma = \left\langle \mathbf{L}_{0,k_0}^{0,l_0}, \cdots, \mathbf{L}_{i_t,k_t}^{j_t,l_t}, \mathbf{L}_{i_{t+1},k_{t+1}}^{j_{t+1},l_{t+1}}, \cdots, \mathbf{L}_{n-1,k_{|\Gamma|-1}}^{n-1,l_{|\Gamma|-1}} \right\rangle, \tag{4}$$

where $i_t \leq i_{t+1}$ and $j_t \leq j_{t+1}$. The set of ruling lines, Γ, defines a piecewise ruled surface, $\mathbf{R}$, and spans all domain, starting from $\mathbf{L}_{0,k_0}^{0,l_0}$ and terminating at $\mathbf{L}_{n-1,k_{|\Gamma|-1}}^{n-1,l_{|\Gamma|-1}}$. $\mathbf{R}$, in turn, should minimize some fitting error metric to the input surface $\mathbf{S}(u, v)$.

The ruled surface constructed in this way is G^1 when $n \rightarrow \infty$. In practice, the 4D-DP creates a dense list of ruling lines, Γ (i.e. Eq. (4)), which is monotone (but not strictly monotone) in both i and j. If a smooth result is desired, one can fit a pair of smooth and regular spline curves to the sequence of corresponding points on Γ. This curve fitting should ensure monotonicity, on one hand, and should bound the error it introduces, on the other.

We now introduce the metrics under which the fitting takes place, and then present methods for evaluating these

metrics. Lastly, the DP algorithm to find the best Γ, given points $\mathbf{P}_{i,k}$ and $\mathbf{Q}_{j,l}$, will be discussed.

2.1. *Metrics for Fitting*

The Hausdorff distance has been widely used to evaluate the shape similarity between two surfaces. However, the computation of the precise Hausdorff distance is typically time-consuming [25] and therefore difficult to use as a metric to search for the best set of ruling line segments.

Here, seeking the best (piecewise) ruling fit of Γ to $\mathbf{S}(u,v)$, we offer two computationally tractable alternatives to [25], one that is based on line-surface distance computations and one that is based on surface-surface distance computation. The latter also provides an upper bound on the precise Hausdorff distance:

(i) **Line-Surface Metric:** The maximal distance from line $\mathbf{L}_{i,k}^{j,l}$ to $\mathbf{S}(u,v)$ is

$$\tilde{D}_H(\mathbf{L}_{i,k}^{j,l}, \mathbf{S}) = \max_{\mathbf{p}\in\mathbf{L}_{i,k}^{j,l}} \min_{u,v} \|\mathbf{p} - \mathbf{S}(u,v)\|, \tag{5}$$

where $\tilde{D}_H$ is the one-sided Hausdorff distance from $\mathbf{L}_{i,k}^{j,l}$ to $\mathbf{S}(u,v)$. Assuming $\mathbf{S}$ is C^1, one can derive an upper bound on $\tilde{D}_H(\mathbf{L}_{i,k}^{j,l}, \mathbf{S})$ by searching for the binormal line to both $\mathbf{L}_{i,k}^{j,l}$ and $\mathbf{S}(u,v)$, a problem that can be formulated as three algebraic constraints in three variables and that can be further reduced to two constraints in two variables by projecting this arrangement to a plane orthogonal to $\mathbf{L}_{i,k}^{j,l}$.

The sought ruled surface should minimize the following line-surface metric in average error:

$$D_{Avg}^{L}(\Gamma, \mathbf{S}) = \frac{\sum_{\mathbf{L}_{i,k}^{j,l}\in\Gamma} A_i^j \tilde{D}_H(\mathbf{L}_{i,k}^{j,l}, \mathbf{S})}{A_S} \tag{6}$$

where A_S is the total area of the input surface $\mathbf{S}$ and A_i^j is the *supporting area* of line $\mathbf{L}_{i,k}^{j,l}$ in $\mathbf{S}$, or the area $\mathbf{L}_{i,k}^{j,l}$ is considered contributing to. A_i^j can be estimated as the area half way between $\mathbf{L}_{i,k}^{j,l}$ and its previous ruling line and half way to the next ruling line. Alternatively, the supporting area of ruling line $\mathbf{L}_{i,k}^{j,l}$ can be estimated as the area of the bilinear patch formed by $\mathbf{L}_{i,k}^{j,l}$ and its previous ruling line, next ruling line, or the average of the two.

As an alternative to the average error we can also employ the L_∞ norm:

$$D_{L_\infty}^{L}(\Gamma, \mathbf{S}) = \max_{\mathbf{L}_{i,k}^{j,l}\in\Gamma} \tilde{D}_H(\mathbf{L}_{i,k}^{j,l}, \mathbf{S}). \tag{7}$$

Assuming $\mathbf{S}$ is C^1 and is Lipschitz continuous, $D_{L_\infty}^{L}(\Gamma, \mathbf{S})$ converges to the one-sided Hausdorff distance from Γ to $\mathbf{S}$ as the distance between two neighboring line segments in Γ is shrinking.

In the ensuing discussion, the computation of *Line Surface Approximate Distance* is denoted by LSAD.

(ii) **Surface-Surface Metric:**

The Line-surface metric is simple to evaluate. However, it is one sided and as a result, regions on $\mathbf{S}$ can be arbitrarily far from the ruling lines, between sampled points. A better, yet somewhat more expensive computationally, symmetric error-bound, that consider surface-surface distances, is now described.

Considering two consecutive ruling line segments $\mathbf{L}_{i_t,k_t}^{j_t,l_t}$ and $\mathbf{L}_{i_{t+1},k_{t+1}}^{j_{t+1},l_{t+1}}$. A bilinear surface patch

$$\mathbf{B}_t = \left\langle \mathbf{P}_{i_t,k_t}, \mathbf{Q}_{j_t,l_t}, \mathbf{Q}_{j_{t+1},l_{t+1}}, \mathbf{P}_{i_{t+1},k_{t+1}} \right\rangle \tag{8}$$

is defined between the four end points[1] of the two ruling segments, $\mathbf{P}_{i_t,k_t}, \mathbf{Q}_{j_t,l_t} \in \mathbf{L}_{i_t,k_t}^{j_t,l_t}$ and $\mathbf{Q}_{j_{t+1},l_{t+1}}, \mathbf{P}_{i_{t+1},k_{t+1}} \in \mathbf{L}_{i_{t+1},k_{t+1}}^{j_{t+1},l_{t+1}}$.

Let $(u_{i_t}, v_{k_t}), (u_{j_t}, v_{l_t}), (u_{j_{t+1}}, v_{l_{t+1}}), (u_{i_{t+1}}, v_{k_{t+1}})$ be the four corresponding parameters of these corners of $\mathbf{B}_t$ in the parametric space of $\mathbf{S}$ and let $\mathbf{b}_t(r,s)$, $r,s \in [0,1]$ be a bilinear surface defined through these four (u,v) parametric locations. Finally, Let $\mathbf{S}_{b_t}$ be the supporting area of $\mathbf{S}$ to $\mathbf{B}_t$, in Euclidean space, as $\mathbf{S}_{b_t} = \mathbf{S}(\mathbf{b}_t(r,s))$. The Hausdorff distance between $\mathbf{B}_t$ and $\mathbf{S}_{b_t}$ is

$$D_H(\mathbf{B}_t, \mathbf{S}_{b_t}) = \max\left(\max_{\mathbf{p}\in\mathbf{B}_t} \min_{r,s} \|\mathbf{p} - \mathbf{S}_{b_t}(r,s)\|, \max_{r,s} \min_{\mathbf{p}\in\mathbf{B}_t} \|\mathbf{p} - \mathbf{S}_{b_t}(r,s)\| \right), \tag{9}$$

and in Section 2.2 we describe how the bound on $D_H(\mathbf{B}_t, \mathbf{S}_{b_t})$ is computed. Here again, one can consider the average error or the L_∞ norms:

$$D_{Avg}^{B}(\Gamma, \mathbf{S}) = \frac{\sum_{\mathbf{B}_t\in\Gamma} A_i^j D_H(\mathbf{B}_t, \mathbf{S}_{b_t})}{A_S}, \tag{10}$$

$$D_{L_\infty}^{B}(\Gamma, \mathbf{S}) = \max_{\mathbf{B}_t\in\Gamma} D_H(\mathbf{B}_t, \mathbf{S}_{b_t}), \tag{11}$$

where A_i^j is the area of patch $\mathbf{S}_{b_t}$ and A_S is the whole area of $\mathbf{S}(u,v)$.

One should note that in Eqs. (6) and (10), only the numerator needs to be evaluated as the denominator, A_S, is constant for a particular input surface $\mathbf{S}$.

2.2. *Surface-Bilinear Surface Hausdorff Distance Bound*

Reconsider the composition $\mathbf{S}(\mathbf{b}_t(r,s))$. This composition extracts the region of $\mathbf{S}$ that is supporting the bilinear region $\mathbf{b}_t(r,s)$. By *support region* we hint to the fact that this region is typically going to be the closest region in $\mathbf{S}$ to $\mathbf{B}_t$. Granted and while unlikely, other, unrelated regions of $\mathbf{S}$ might be closer, but this support region provides an upper bound that is expected to be fairly tight. Further, the supporting region follows the parameterization of $\mathbf{b}_t(r,s)$. A simple scheme to compute the surface-surface composition, in the Bézier domain, can be found in Appendix.

[1] In fact, the final ruled surface obtained from Γ can be approximated in the discrete form by the collection of these bilinear patches.

We now have two surfaces in Euclidean space: surface $\mathbf{S}_{b_t}(r,s)$ that is confined to the bilinear domain $\mathbf{b}_t(r,s)$ and a bilinear surface patch $\mathbf{B}_t(r,s)$ (i.e., Eq. (8)). Further, both surfaces share the $[0,1]^2$ domain but do not share the same functional space (i.e. orders and knot sequences of B-spline surfaces). However, one can apply degree elevation to the surface with the lower orders (i.e., the bilinear surface patch $\mathbf{B}_t$) and insert missing knots as necessary, to make the functional space compatible. Then,

$$\begin{aligned} D_H(\mathbf{B}_t, \mathbf{S}_{b_t}) &\leq \max_{r,s} \|\mathbf{S}_{b_t}(r,s) - \mathbf{B}_t(r,s)\| \\ &= \max_{r,s} \left\| \sum_{i=0}^{k_u} \sum_{j=0}^{k_v} \mathbf{P}_{ij}^{S_{b_t}} B_i(r) B_j(s) - \right. \\ &\qquad \left. \sum_{i=0}^{k_u} \sum_{j=0}^{k_v} \mathbf{P}_{ij}^{B_t} B_i(r) B_j(s) \right\| \\ &= \max_{r,s} \left\| \sum_{i=0}^{k_u} \sum_{j=0}^{k_v} \left(\mathbf{P}_{ij}^{S_{b_t}} - \mathbf{P}_{ij}^{B_t} \right) B_i(r) B_j(s) \right\| \\ &\leq \max_{i,j} \left\{ \|\mathbf{P}_{ij}^{S_{b_t}} - \mathbf{P}_{ij}^{B_t}\| \right\}, \end{aligned} \tag{12}$$

where $\mathbf{P}_{ij}^{S_{b_t}}$ and $\mathbf{P}_{ij}^{B_t}$ are the control points of $\mathbf{S}_{b_t}$ and $\mathbf{B}_t$, respectively, and k_u and k_v are the respective lengths of the control meshes of the surfaces. By this formulation, one should simply examine the maximum difference between corresponding control points of the two surfaces for the error-bound of $D_H(\mathbf{B}_t, \mathbf{S}_{b_t})$. Hence after, the computation of the *Surface-Surface Hausdorff-distance Bound* following Eq. (12) is denoted as SSHB.

When degree raising (or refining) the bilinear surface $\mathbf{B}_t$, two options can be used - either place the new control points uniformly in $\mathbf{B}_t$ or let the control points of $\mathbf{B}_t$ proportionally distributed according to the control points of $\mathbf{S}_{b_t}$, a method we denote *projected parameterization* (see Fig. 3 for an illustration). Specifically, the control points of $\mathbf{B}_t$ are

$$\mathbf{P}_{ij}^{B_t} = \begin{bmatrix} 1 - \frac{i}{k_u} & \frac{i}{k_u} \end{bmatrix} \begin{bmatrix} \mathbf{P}_{i_t,k_t} & \mathbf{Q}_{j_t,l_t} \\ \mathbf{P}_{i_{t+1},k_{t+1}} & \mathbf{Q}_{j_{t+1},l_{t+1}} \end{bmatrix} \begin{bmatrix} 1 - \alpha_j \\ \alpha_j \end{bmatrix} \tag{13}$$

with $\alpha_j = \frac{j}{k_v}$ in the uniform parameterization. When taking projected parameterization on $\mathbf{B}_t$,

$$\alpha_{j\ (j\neq 0)} = \frac{\sum_a^j \|\hat{\mathbf{P}}_{0,a-1}^{S_{b_t}} \hat{\mathbf{P}}_{0,a}^{S_{b_t}}\|}{\sum_a^{k_v} \|\hat{\mathbf{P}}_{0,a-1}^{S_{b_t}} \hat{\mathbf{P}}_{0,a}^{S_{b_t}}\|}, \quad \alpha_0 = 0, \tag{14}$$

where $\hat{\mathbf{P}}_{0,j}^{S_{b_t}}$ is the projected point of $\mathbf{S}_{b_t}$'s control point, $\mathbf{P}_{0,j}^{S_{b_t}}$, on the line $\mathbf{P}_{i_t,k_t}\mathbf{Q}_{j_t,l_t}$. Note that, α_j defined in this way is monotonic and $\alpha_j \in (0,1)$, and the shape of $\mathbf{B}_t(u,v)$ will keep as a bilinear patch. While *projected parameterization* is somewhat more expensive computationally, it also provides a tighter bound on $D_H(\mathbf{B}_t, \mathbf{S}_{b_t})$ as will be demonstrated in the results section.

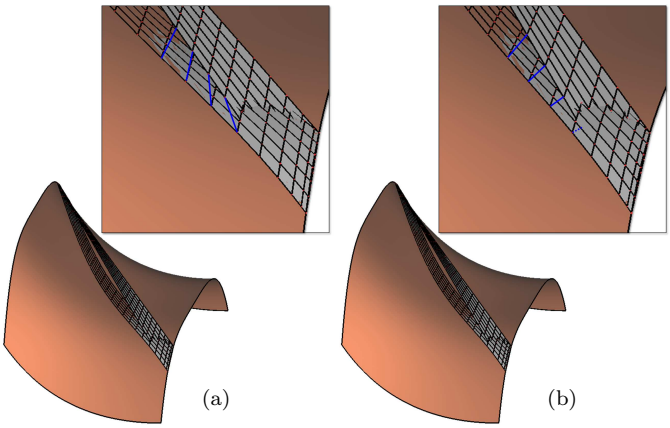


Fig. 3. Tightening the error-bound by *projected parameterization* (See Section 2.2): (a) the control mesh of $\mathbf{B}_t$ is computed using uniform parameterization and (b) the distribution of $\mathbf{B}_t$'s control points is adjusted to follow $\mathbf{P}_{i,j}^{S_{b_t}}$, the control points of $\mathbf{S}_{b_t}$. To highlight the difference on the distribution of control points on $\mathbf{B}_t$, the correspondence between $\mathbf{P}_{ij}^{S_{b_t}}$s and $\mathbf{P}_{ij}^{B_t}$s are illustrated by blue lines in the zoom views. In this example, when using projected parameterization, the error of the computed Hausdorff distance bound is reduced by two orders of magnitude.

2.3. *Best Ruling Fit*

Having the ability to bound $D_H(\mathbf{B}_t, \mathbf{S}_{b_t})$ (i.e., Eq. (9)), we are now ready to derive the optimal ruling lines for approximating $\mathbf{S}$. Assume, for now, that $\mathbf{S}$ is open and hence the first ruling line must be between $\mathbf{P}_{0,k_0}$ and $\mathbf{Q}_{0,l_0}$ (i.e., $\mathbf{L}_{0,k_0}^{0,l_0}$), where all elevated combinations of k_0 and l_0 should be examined as initial candidates and selecting the best fit. Similarly, recalling we sampled n samples in $\mathbf{C}_0$ and $\mathbf{C}_1$, the last ruling line must be $\mathbf{L}_{n-1,k_{|\Gamma|-1}}^{n-1,l_{|\Gamma|-1}}$. Considering an intermediate ruling line $\mathbf{L}_{i_t,k_t}^{j_t,l_t}$, the next ruling line can be obtained by succeeding in

(i) a P-point only (i.e., $i_{t+1} = i_t + 1$ and $j_{t+1} = j_t$),
(ii) a Q-point only (i.e., $i_{t+1} = i_t$ and $j_{t+1} = j_t + 1$), or
(iii) both P- and Q-points (i.e., $i_{t+1} = i_t + 1$ and $j_{t+1} = j_t + 1$).

Each such point succession has three choices of moving

(i) upward, in the normal direction, by assigning $k_{t+1} = k_t + 1$ (or $l_{t+1} = l_t + 1$),
(ii) downward, in the reversed normal direction, by assigning $k_{t+1} = k_t - 1$ (or $l_{t+1} = l_t - 1$), or
(iii) staying at the same elevation by keeping k_t (or l_t) unchanged.

Assume the successor line of $\mathbf{L}_{i_t,k_t}^{j_t,l_t}$ is line $\mathbf{L}_{i_{t+1},k_{t+1}}^{j_{t+1},l_{t+1}}$. Then, the approximation error between the bilinear patch formed by $\mathbf{L}_{i_t,k_t}^{j_t,l_t}\mathbf{L}_{i_{t+1},k_{t+1}}^{j_{t+1},l_{t+1}}$ and $\mathbf{S}$ can be computed by SSHB, introduced in Section 2.1. The supporting area of the bilinear patch $\mathbf{L}_{i_t,k_t}^{j_t,l_t}\mathbf{L}_{i_{t+1},k_{t+1}}^{j_{t+1},l_{t+1}}$ is approximated by the summed area of two triangles: $\triangle\mathbf{P}_{i_t,k_t}\mathbf{Q}_{j_{t+1},l_{t+1}}\mathbf{Q}_{j_t,l_t}$ and $\triangle\mathbf{P}_{i_t,k_t}\mathbf{Q}_{j_{t+1},l_{t+1}}\mathbf{P}_{i_{t+1},k_{t+1}}$, or a refined tessellation on the given surface $\mathbf{S}(u,v)$. By this formulation, the optimal set of ruling lines, which minimize the metrics $D_{Avg}^B(\Gamma, \mathbf{S})$, are

determined by dynamic-programming [26], in two phases:

Phase I of DP: Scoring

To compute the minimal averaged-error of bilinear patches to $\mathbf{S}$ in some path from $\mathbf{L}_{0,k_0}^{0,l_0}$ to $\mathbf{L}_{n-1,k_{|\Gamma|-1}}^{n-1,l_{|\Gamma|-1}}$, we initialize a 4D *score table* for the *dynamic-programming* (DP) with dimensions: $n \times n \times (2m+1) \times (2m+1)$.

Consider an intermediate ruling line $\mathbf{L}_{i,k}^{j,l}$, and denote its (so-far) score value by $D_{Avg}^{i,j,k,l}$. For this averaging case, and considering the predecessors of $D_{Avg}^{i,j,k,l}$, the score table entries are updated by the following rule:

$$D_{Avg}^{i,j,k,l} = \min_{j_t,\, l_t,\, i_t,\, k_t} \left\{ D_{Avg}^{i_t,j_t,k_t,l_t} + D_H(\mathbf{L}_{i,k}^{j,l}\mathbf{L}_{i_t,k_t}^{j_t,l_t}, \mathbf{S}) \right\}, \quad (15)$$

with

$$\begin{aligned} (i_t,k_t) &= \{(i,k),(i-1,k),(i-1,k-1),(i-1,k+1)\}, \\ (j_t,l_t) &= \{(j,l),(j-1,l),(j-1,l-1),(j-1,l+1)\}, \\ 0 \le i_t, j_t, &< n \quad -m \le k_t, l_t \le m. \end{aligned} \quad (16)$$

To avoid singularities and self-intersections in the DP computation, we do not allow P- or Q-points to move *upward* or *downward* without moving *forward* (i.e. the predecessor options of $(i, k \pm 1)$ and $(j, l \pm 1)$ are excluded). Only *straight-forward*, *up-forward*, *down-forward* or *stay stationary* moves are allowed). Furthermore, stationary movies of both $(i_t, k_t) = (i, k)$ and $(j_t, l_t) = (j, l)$ cannot be selected simultaneously. In summary, Eq. (16), offers 15 different possible movements of the P-Q ruling lines.

Similarly, for minimizing the maximal error case, we have (compare with Eq. (15)):

$$D_{L_\infty}^{i,j,k,l} = \min_{j_t,\, l_t,\, i_t,\, k_t} \left\{ D_{L_\infty}^{i_t,j_t,k_t,l_t}, D_H(\mathbf{L}_{i,k}^{j,l}\mathbf{L}_{i_t,k_t}^{j_t,l_t}, \mathbf{S}) \right\} \quad (17)$$

with the same set of candidate combinations of (i_t, k_t) and (j_t, l_t) as in Eq. (16).

All the entries of the 4D table can be updated row by row, top to bottom, left to right in four nested loops – i.e., letting $i, j = 0 \Rightarrow n-1$ and $k, l = -m \Rightarrow m$. The computational complexity of this update is in the order of the number of entries in the table, or $O(n^2m^2)$.

Finally, as a standard step of DP, every entry (i, j, k, l) in the 4D score table should be record which predecessor is used to update the score at (i, j, k, l), so we can back trace it.

Phase II of DP: Back-tracking

Once all the scores of all the entries in the 4D table are computed, the optimal solution path can be determined by the following two steps:

- Checking all the entries with $i = j = n-1$ to find an entry $(n-1, n-1, k_o, l_o)$ with minimal value $D_{Avg}^{n-1,n-1,k_o,l_o} = \min_{k,l}\{D_{Avg}^{n-1,n-1,k,l}\}$. If there are more than one such end ruling line with minimal score value, we can randomly pick one or adopt some heuristic to choose one (e.g., selecting the ruling line with $\min\{|k| + |l|\}$ which is less elevated from the given surface).
- Starting from the selected end ruling line, and using the recorded predecessor of each entry, the sought sequence of ruling lines can be traced back one by one. The process of back-tracking stops when reaching an entry with $i = j = 0$.

At every step, either i or j (or both), are decreased in the back-tracking procedure that hence always terminates. The piecewise ruled surface that globally minimizes the desired discrete norm for $\mathbf{S}$ can hence be created in $O(n+m)$.

2.4. *Fitting Periodic Surfaces*

The above multi-dimensional DP based method for searching for the best ruled surface in the discrete domain can be extended to fit periodic surfaces as well. Finding the best periodic ruled surface can be realized by a 5D dynamic-programming with dimensions: $n \times n \times (2m+1) \times (2m+1) \times n$, where we introduce an additional dimension of size n to consider shifting the starting ruling line from $\mathbf{L}_{0,k}^{0,l}$ to $\mathbf{L}_{0,k}^{q,l}$ along this new dimension (denoted the Q-shifting dimension). The update rules for the score table entries are similar to what are introduced above. Specifically, the value of table entry $D_{Avg}^{i,j,k,l,q}$ can only be updated by the other entries with the same q shift. From Eq. (15), we obtain

$$\begin{aligned} D_{Avg}^{i,j,k,l,q} = \min\{ & D_{Avg}^{i_t,j_t,k_t,l_t,q} + \\ & D_H(\mathbf{L}_{i,k}^{(j+q) \bmod n,l}\mathbf{L}_{i_t,k_t}^{(j_t+q) \bmod n,l_t}, \mathbf{S})\}. \end{aligned} \quad (18)$$

As a result, we do not really need to run a 5D-DP. Instead, we conduct n 4D-DP's by using different shift values of $q \in \{0, 1, \cdots, n-1\}$. Similarly, for $D_{L_\infty}^{i,j,k,l,q}$ and following Eq. (17), we get

$$\begin{aligned} D_{L_\infty}^{i,j,k,l,q} = \min\{ & D_{L_\infty}^{i_t,j_t,k_t,l_t,q}, \\ & D_H(\mathbf{L}_{i,k}^{(j+q) \bmod n,l}\mathbf{L}_{i_t,k_t}^{(j_t+q) \bmod n,l_t}, \mathbf{S})\}. \end{aligned} \quad (19)$$

One difficulty in using the above DP to fit a periodic ruled surfaces stems from the fact that the starting ruling line and the ending ruling line must coincide (see Fig. 4 for an example). We enforce the coincidence of $\mathbf{L}_{0,k_0}^{0,l_0}$ and $\mathbf{L}_{n-1,k_{|\Gamma|-1}}^{n-1,l_{|\Gamma|-1}}$ by modifying the score updating rule as follows:

- At the initialization step, the scores of all $D_{Avg\ /\ L_\infty}^{n-1,n-1,k,l,q}$ are assigned to infinity.
- During the score update, not only the predecessors but also the 'source' of the first ruling line $\mathbf{L}_{0,k_0}^{q,l_0}$ which contributes to the score $D^{i,j,k,l,q}$ is recorded in each entry of the scoring table.
- When the updating process gets to the end with $i = n-1$ and $j = n-1$, the score updating of $D_{Avg\ /\ L_\infty}^{n-1,n-1,k,l,q}$ is only allowed if the 'source' of its predecessor is at the same level in normal elevation (i.e., $k = k_s$ and $l = l_s$ when the 'source' of its predecessor is $\mathbf{L}_{q,k_s}^{0,l_s}$).

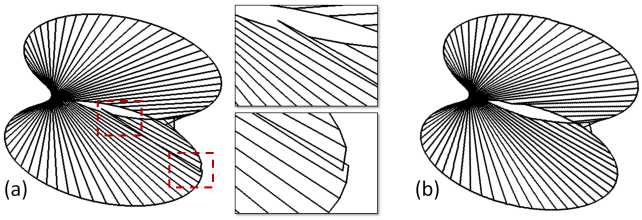


Fig. 4. Fitting a periodic surface: (a) the 4D-DP result without enforcing the coincidence of starting and ending ruling lines – gaps are generated at the places highlighted by dashed red squared (and scaled up), and (b) the result after applying the coincident constraint.

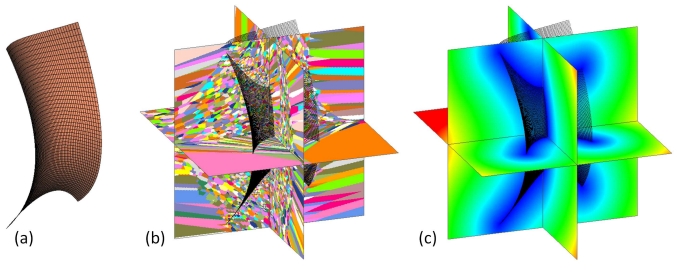


Fig. 5. The discrete VD (res.: 512^3) of a cubic NURBS surface: (a) the input NURBS surface, (b) the discrete VD where the regions have the same site are displayed in the same color, and (c) the corresponding distance-field of VD. Shown are main sectional planes of the different fields.

3. GPU-based Algorithms for Efficient Computing

We now present the GPU-based algorithms for computing the ruled surface approximating to a given surface $\mathbf{S}$ in the discrete domain. First, a method for computing SSHB is introduced, which can be parallelized and run on GPUs. Second, we develop a method to conduct the approximate distance queries between a ruling line and $\mathbf{S}$ in a more efficient way with the help of distance-field on GPUs.

In practice, SSHB was mainly used in 4D-DP, and LSAD in higher dimensions DP (e.g., 5D-DP for fitting periodic surfaces), mainly due to the excessive computing and memory needs of the SSHB method.

3.1. *GPU/CPU computation for SSHB*

The surface-surface distance computation with a tight error-bound is time-consuming. For a 4D-DP with dimension $50^2 \times 41^2$, it takes more than an hour on a PC with iCore 7 CPU. Most of the computation time is spent on the surface-surface distance query (i.e., the step that fills the score table of the DP). To reduce the computing time, a GPU/CPU hybrid scheme is developed to compute the SSHB for the score table used in the DP.

At first, we compute the set of control meshes for all possible surface regions on the given surface $\mathbf{S}(u, v)$ according to the boundary samples $\mathbf{C}_0(u_i)$ and $\mathbf{C}_1(u_j)$. As $i, j \in \{0, \cdots, n-1\}$, there are

(i) $(n-1) \times n$ triangular patches formed by $\mathbf{C}_0(u_i)$, $\mathbf{C}_0(u_{i+1})$ and $\mathbf{C}_1(u_j)$ (called *P-succeed* patches),
(ii) $n \times (n-1)$ triangular patches formed by $\mathbf{C}_0(u_i)$, $\mathbf{C}_1(u_j)$ and $\mathbf{C}_1(u_{j+1})$ (called *Q-succeed* patches),
(iii) and $(n-1) \times (n-1)$ quadrangular patches formed by $\mathbf{C}_0(u_i)$, $\mathbf{C}_0(u_{i+1})$, $\mathbf{C}_1(u_{j+1})$ and $\mathbf{C}_1(u_j)$ (called *PQ-succeed* patches).

The control points of all *P*-succeed, *Q*-succeed and *PQ*-succeed patches are computed at the CPU side with the help of IRIT library [27] and stored in three arrays.

The three arrays are copied to the memory of the GPU, and the solution of a 4D-DP is computed by a CPU/GPU hybrid approach. According to the update rules of the score table (e.g., either Eq. (15) or Eq. (17)), we construct 15 surface-surface distance tables (in 4D) – according to the 15 possible *P*- and *Q*-succeeding options presented in Eq. (16). The entries of these tables are filled in a highly parallel manner. Specifically, the 15 surface-surface distance tables, $\Theta_{1,\cdots,15}(i, j, k, l)$, are for the items, $D_H(\mathbf{L}^{j_t,l_t}_{i_t,k_t}\mathbf{L}^{j,l}_{i,k}, \mathbf{S})$ (with (i_t, k_t) and (j_t, l_t) as given in Eq. (16)), that are checked during the 4D-DP score table update. Each thread fills one entry of the score table by computing the corresponding distance between the bilinear patch formed by the two ruling lines $\mathbf{B}_t =< \mathbf{L}_t\mathbf{L}_{t+1} >$ and its corresponding surface region $\mathbf{S}_{b_t}$. Specifically, the control points of $\mathbf{B}_t$ are refined to follow the surface $\mathbf{S}_{b_t}$, by a projected parameterization (see Section 2.2). An error-bounded distance is then established by Eq. (12). Both steps are implemented in a highly parallel way running on GPUs.

Copying the entries of $\Theta_{1,\cdots,15}(i, j, k, l)$ back to the CPU side, the score table of 4D-DP can be updated according to Eqs. (15) or (17). Lastly, the best ruling lines according to the scores can be determined at the CPU side by the back-tracking method proposed in Section 2.3. For the 4D-DP with a table of size $50^2 \times 41^2$, this GPU/CPU hybrid approach can generate the optimal fitting in around 5 seconds.

3.2. *Approximate distance queries on GPUs*

One drawback of the above method is that the surface-surface distance tables can easily exhaust the graphics memory when the dimension of DP or the sampling rate (i.e., n and m) increases. To realize the computation on high dimensional DP and dense samples, the line-surface distance based metrices can be adopted (i.e., Eqs. (6) and (7)). In the update rules of the 4D-DP, the error term introduced by a bilinear patch (Eq. (9)) are changed to the error introduced by a new ruling line (Eq. (5)). By this change, a more efficient but less accurate (and no error bound) GPU-based algorithm can be realized (i.e., LSAD).

A discrete *Voronoi* diagram (VD) sampled on uniform voxels using a GPU-based distance transformation [28] is constructed to help conducting line-surface distance queries to the given freeform surface $\mathbf{S}(u, v)$. The VD employed here is a discrete one. The voxels containing any point of $\mathbf{S}(u, v)$ are labeled as *site* points of VD. Every voxel stores the coordinate of its nearest site point in the discrete domain $\mathbb{Z}^3$ of voxels, specifically for a voxel (i, j, k) the value $I(i, j, k) \in \mathbb{Z}^3$ gives the coordinates of its nearest site. Figure 5 shows the example VD of a NURBS surface.

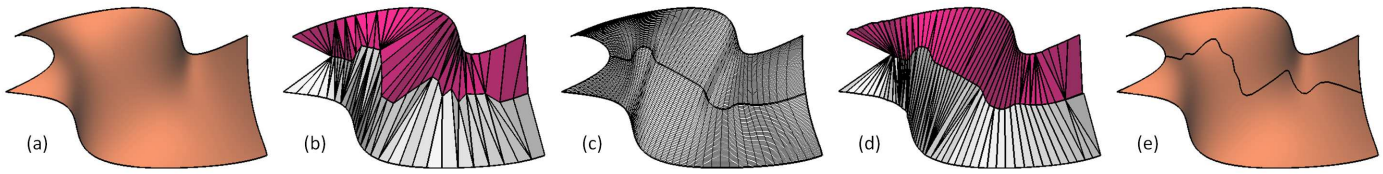


Fig. 6. Subdivision by optimal fitting of ruled surfaces: (a) a given Bézier surface patch with order 9 and 6 in u, v-directions, (b) the ruling line segments of 4D-DP composite fitting with dimensions: $30^3 \times 10$, (c) the mesh of subdivided surface patches obtained by 4D-DP composite fitting, (d) the fitting result by applying our ruling line approximation (4D-DP with dimensions: $50^2 \times 11^2$) on two sub-surfaces, and (e) subdivided surface patches obtained by 7D-DP composite fitting with dimensions: $(30 \times 11)^3 \times 10$.

To evaluate the distance between a line segment $\mathbf{q}_s\mathbf{q}_e$ and $\mathbf{S}(u, v)$ (with $\mathbf{q}_s$ and $\mathbf{q}_e \in \Re^3$), we need to first find out all the voxels (i_t, j_t, k_t) that are swept by a point $\mathbf{q}(t) = (1-t)\mathbf{q}_s + t\mathbf{q}_e$ with $t \in [0, 1]$. Then, the closest distance between the centers of these voxels and $\mathbf{q}_s\mathbf{q}_e$ is used as an approximation of the one-side Hausdorff distance between $\mathbf{q}_s\mathbf{q}_e$ and $\mathbf{S}(u, v)$, which can be efficiently computed in parallel on GPUs.

4. Optimizing the Surface Subdivision in a Discrete Ruled Surface Fitting

The dynamic programming based method for computing optimal ruled surface fitting can be further extended to determine an optimal ruling subdivision curve for $\mathbf{S}$ – i.e., the RSP problem. The solution domain is discretized in a higher dimension. The boundary curves $\mathbf{C}_0(u)$ and $\mathbf{C}_1(u)$ of $\mathbf{S}(u, v)$ are uniformly sampled into n P- and n Q-points as in Eq. (1). However, the interior region of surface $\mathbf{S}$ is now also sampled by $n \times d$ points as

$$\begin{aligned}\mathbf{M}_{a,b} = \mathbf{S}((1 - \tfrac{a}{n-1})u_{\min} + \tfrac{a}{n-1}u_{\max}, \\ (1 - \tfrac{b+1}{d+1})v_{\min} + \tfrac{b+1}{d+1}v_{\max})\end{aligned} \tag{20}$$

with $a \in [0, n-1]$ and $b \in [0, d-1]$. These new points are denoted M-points. By this formulation, picking one P-point, one Q-point and one M-point can form a pair of ruling line segments: $\mathbf{L}_{i,(a,b)} \equiv \mathbf{P}_i\mathbf{M}_{a,b}$ and $\mathbf{L}_{j,(a,b)} \equiv \mathbf{M}_{a,b}\mathbf{Q}_j$. The RSP problem is reduced to finding a sequence of such pairs of ruling line segments. For a set of ruling line pairs, Υ, the solution of RSP problem needs to minimize the difference between the ruled surfaces and $\mathbf{S}(u, v)$, where the ruled surfaces are approximated by $\{\mathbf{L}_{i,(a,b)}\}$ and $\{\mathbf{L}_{j,(a,b)}\}$.

Again, to avoid self-intersections in the DP computation, we allow P- or Q-points to either move *forward* or keep their current position. For an M-point, it can move *left-forward*, *straight-forward*, *right-forward* or stay *stationary*. Finally, it is not allowed to keep all three points stationary.

The optimal M-curve in the form of a sequence of M-points can be determined by the dynamic programming framework. If none of the P-, Q- and M-points is allowed to be elevated above/below $\mathbf{S}(u, v)$, the result can be obtained by a 4D-DP. If only M-points can be elevated along surface normals, a 5D-DP is used. The dimension of DP can be further raised to 7D if all P-, Q- and M-points are allowed to be elevated (see Fig. 6 for an example).

Virtual refinement: The subdivision of a given surface $\mathbf{S}(u, v)$ by a general interior curve $\mathbf{b}_M(t) = (u(t), v(t))$ into two surface patches $\mathbf{S}_L$ and $\mathbf{S}_R$ that together precisely represent the original shape of $\mathbf{S}$ is not trivial and typically necessitates the use of surface-surface composition. Moreover, such a subdivision will increase the complexity of the resulting surfaces (e.g., raise the degrees due to the composition and hence the number of control points). Further recursive subdivision of the surfaces, toward a finer ruled surfaces' fitting (as is shown in Fig. 11), expands the magnitude of this degree raising difficulty. Hence, we adopt a strategy of *virtual subdivision* to overcome this problem.

For a given tensor surface $\mathbf{S}(u, v)$, its four boundary curves are represented by four straight lines, in the parametric domain. Assume, w.l.o.g., the initial subdivision curve $\mathbf{b}_M(t) = (u(t), v(t))$ according to the M-points in the u, v-domain spans from the $u = 0$ boundary to the $u = 1$ boundary. The two new subdivided surface patches $\mathbf{S}_L(u^l, v^l), \forall u^l, v^l \in [0, 1]$ and $\mathbf{S}_R(u^r, v^r), \forall u^r, v^r \in [0, 1]$ can be represented as follows:

(i) A continuous mapping Ω_L from $u^l, v^l \in [0, 1]$ to the u, v region which is bounded by $\{u = 0, u = 1, v = 0, \mathbf{b}_M(t)\}$ is constructed with the help of a Coons patch interpolation [30]. Similarly, a mapping Ω_R from $u^r, v^r \in [0, 1]$ to the u, v region which is bounded by $\{u = 0, u = 1, \mathbf{b}_M(t), v = 1\}$ is formulated as another Coons patch interpolation.

(ii) When using the method in Section 2 to evaluate the shape error, every point in the u^l, v^l (or u^r, v^r) domain is mapped to a point in the u, v domain by Ω_L or Ω_R, evaluating the surface-surface composition on a point-by-point basis.

5. Results

This section highlight the performance of our algorithm on different examples. All the results reported here were generated on a PC using an Intel Core i7 3.10GHz CPU + 8GB RAM and a GeForce GTX580 GPU with 3GB of video memory. We implemented our algorithms in C++ with the help of Visual Studio 2008, the IRIT [27] library, and the nVidia CUDA SDK.

The first two examples shown in Fig. 1 and Fig. 7 are two open Bézier surfaces, which are quite well fitted with a ruling approximation as these input surfaces are hyperbolic. The statistics in terms of fitting errors, the average error – $D^B_{L_2}$ (Eq. (10)) and the maximal error – $D^B_{L_\infty}$ (Eq. (11)), are

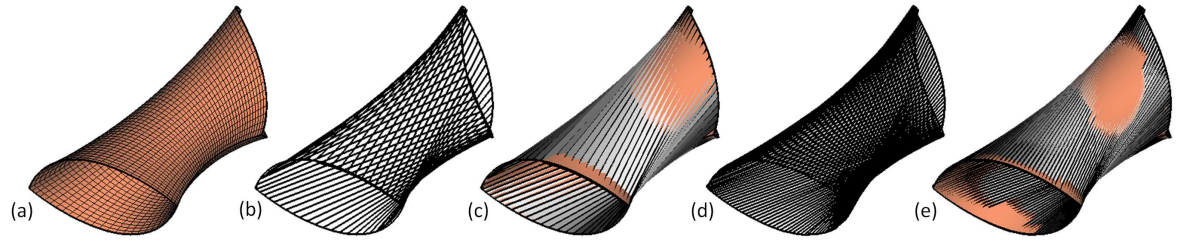


Fig. 10. The ruling approximation of a periodic B-spline surface (a). The result by 5D-DP with dimensions: $50^3 \times 10^2$ is shown in (b) and (c), and the result by 5D-DP with dimensions: $100^3 \times 10^2$ is shown in (d) and (e).

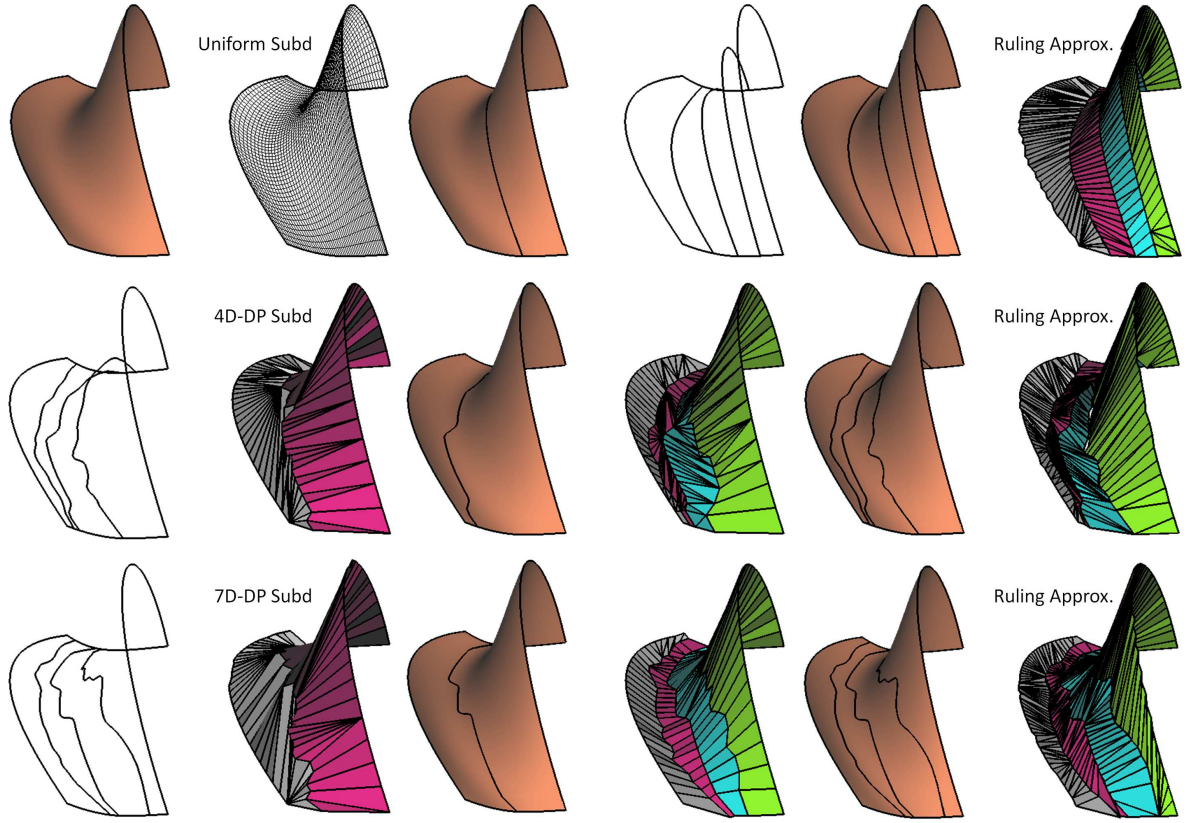


Fig. 11. The subdivision of a surface using different methods: (top row) uniform refinement by iso-parametric curves $v = 0.25, 0.5, 0.75$, (middle row) refinement at M-curves determined by 4D-DP based optimal fitting (res.: $30 \times 30 \times 30 \times 10$), and (bottom row) refinement at curves determined by 7D-DP based optimal fitting (res.: $(30 \times 5)^3 \times 10$). The last column shows the results of ruling approximation on the refined surface patches (4D-DP res.: $50^2 \times 11^2$).

shown in Fig. 8, where the optimal fitting results by 4D-DP are compared with the results of 2D-DP by using the benchmark with different sampling rates (i.e., $n = 30, 50, 100$). In addition and for further assurance, the maximal errors ($E_{\max}$) and the mean errors (E_{mean}) between the surfaces are estimated with the help of the publicly available Metro tool [32] – see Fig. 9. When 4D-DP is conducted (rather than 2D-DP), the maximal error $E_{\max}$ and the mean error E_{mean} can drop up to 40% and 35% respectively.

The example shown in Fig. 4 is a periodic surface. Another example of a periodic surface is shown in Fig. 10. Our fitting results present maximal errors: $E_{\max} = 0.0193\bar{L}$ in the example of Fig. 4) and $E_{\max} = 0.0109\bar{L}$ in the example of Fig. 10), where $\bar{L}$ denotes the diagonal length of input surface's bounding box. Again, the errors are estimated by the third party software – the Metro tool [32].

Figure 6 shows the results of applying the subdivision based on optimal fitting using 4D-DP. An example for the comparison of using 4D-DP and 7D-DP in determining the optimal subdivision is given in Fig. 11. Fig. 12 gives the error comparison on the fitting results obtained on the 4D-DP/7D-DP based subdivisions vs. the uniform subdivision (i.e., subdivision on iso-parametric curves $v = 0.25, 0.5, 0.75$ – see the top row of Fig. 11). The fitting results are all generated by 4D-DP with dimensions: $50^2 \times 11^2$. Subdivision based on 4D-DP gives much better results than the uniform subdivisions. The subdivision curve determined by 7D-DP is only slightly better than 4D-DP (in terms of maximal errors); however, 7D-DP has a much higher consumption of both memory and computing power

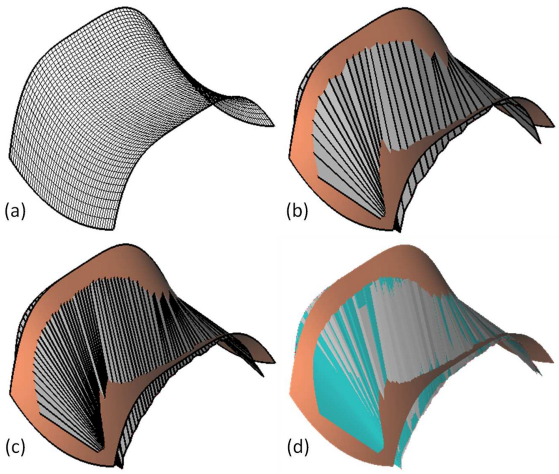
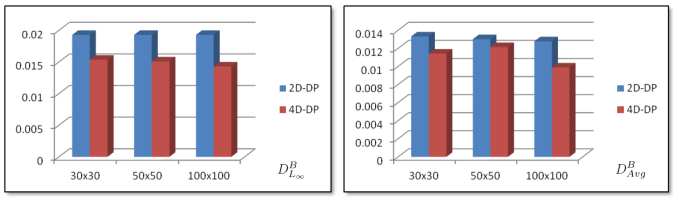
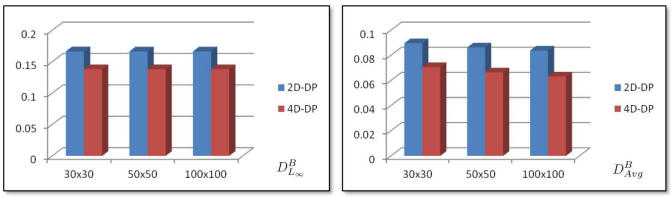


Fig. 7. The ruling approximation of an open Bézier surface (a). Computations using different sampling rate in 4D-DP converges to a similar shape – (b) 25^4, (c) 50^4, and (d) 100^4 (in cyan) which is displayed together with the result of 50^4 (in gray).



(a) Example I – Fig. 1



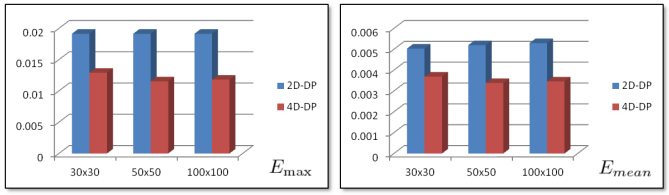
(b) Example II – Fig. 7

Fig. 8. Error analysis on the fitting results of ruling approximation by SSHB: 2D-DP (Dim.: $n \times n$) vs. 4D-DP (Dim.: $n \times n \times 41 \times 41$) with $n = 30, 50, 100$. The average errors (D^B_{Avg}) and the maximal errors ($D^B_{L_\infty}$) are evaluated by Eqs. (10) and (11) respectively.

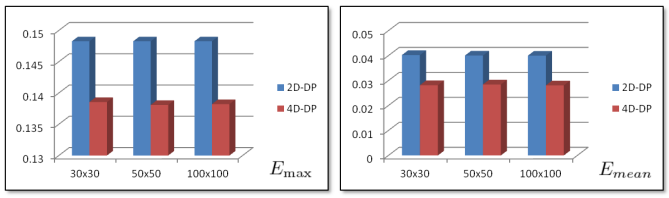
(see Table 3). During the computation of optimal subdivision, the maximal error, $D^L_{L_\infty}(\Gamma, \mathbf{S})$ (Eq. (7)), is used. As a result, mean errors on the results of 7D-DP subdivision could be slightly larger than 4D-DP subdivision.

The last example (see Fig. 13) is employed to demonstrate the expected benefits of the projected parameterization. The rulings result obtained by 4D-DP gives inferior approximation when the uniform parameterization is used compared to the projected parameterization.

In our algorithm, the control meshes of sub-surface patches are generated with the help of the IRIT library. For the cases with $n < 100$, the computation can be finished in 2 seconds by using only one CPU. Table 2 lists the statistics on the example shown in Fig. 1 when using different resolutions in both SSHB 4D-DP and LSAD 4D-DP. The LSAD 4D-DP is more efficient as it can be fully evaluated on GPU with the help of discrete Voronoi diagram.



(a) Example I – Fig. 1



(b) Example II – Fig. 7

Fig. 9. Error analysis on the fitting results of ruling approximation by SSHB: 2D-DP (Dim.: $n \times n$) vs. 4D-DP (Dim.: $n \times n \times 41 \times 41$) with $n = 30, 50, 100$. The maximal errors ($E_{\max}$) and the mean errors (E_{mean}) between two surfaces are estimated with the help of the publicly available Metro tool [32].

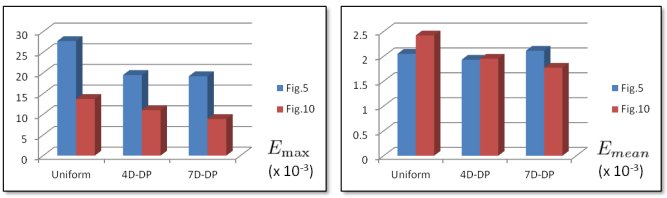


Fig. 12. Error comparison for different subdivision methods on two examples in Figs.6 and 11 – uniform subdivision, optimal subdivision by 4D-DP and optimal subdivision by 7D-DP. The maximal errors ($E_{\max}$) and the mean errors (E_{mean}) between the input surfaces and the fitting results (by 4D-DP with dimension: $50^2 \times 11^2$) on subdivided patches are evaluated with the help of the publicly available Metro tool [32].

Table 2
Statistics of Computational Time (ms) on SSHB 4D-DP and LSAD 4D-DP with Different Dimensions.

Dimensions: $m^2 \times 49^2$	$m = 50$	$m = 60$	$m = 70$	$m = 80$
Generation of Control-Meshes	312	468	671	967
Surf-Surf Distance Evaluations	5,320	7,691	10,530	13,853
4D-DP Score Updating	811	1,076	1,513	2,137
Total Time† (SSHB)	6,474	9,282	12,777	17,035
Line-Surf Distances Evaluation	141	203	296	375
4D-DP Score Updating	749	952	1436	1,763
Total Time‡ (LSAD)	905	1,201	1,763	2,184

† The total time includes the back-tracing time and the time for memory allocation and release.
‡ The total time includes the generation of discrete Voronoi diagram, the back-tracing and the memory management.

Statistics of computational time on a periodic surface (the example in Fig. 4 by using LSAD) is given in Table 4.

Table 3
Statistics of the Optimal Subdivisions in Fig. 11.

Strategies	4D-DP	7D-DP
Dimensions	$30 \times 30 \times 30 \times 10$	$(30 \times 5)^3 \times 10$
Time (ms)	2,993	7,530
CPU Memory (MB)	3.09	386
GPU Memory (MB)	12.1	523

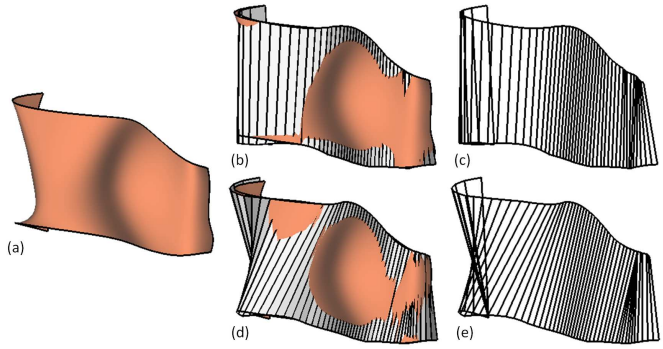


Fig. 13. The tightness of the error-bound in SSHB can be improved by using projected parameterization (see Section 2.2). For a given surface (a), the uniform parameterization leads to a loose error-bound so that the fitting result does not give a good shape approximation (see (b) and (c)) – by 4D-DP (dimensions: $50^2 \times 11^2$). The error-bound can be tightened by using the projected parameterization – see the results using 4D-DP with the same dimension in (d) and (e). In this case, the Hausdorff distance between the input surface and the fitting result is reduced by 15%.

6. Conclusion and Discussion

In this work, we presented a method to compute an optimal set of ruling lines to solve the discrete RSF problem. Multi-dimensional dynamic programming is conducted to find the global optimum in a discrete domain. We sampled the boundary curves of a given surface and elevate them along surface normals. The discrete domain has four degrees-of-freeform for solving the RSF problem and is formulated over these samples.

In our current implementation, the discrete domains are formed by the sample points on $\mathbf{S}(u, v_{\min})$ and $\mathbf{S}(u, v_{\max})$ and the points evaluated along surface normals on these two curves. One may wish to generate the discrete domain on other iso-parametric curves (e.g., $\mathbf{S}(u, v_{\min} + \Delta)$ and $\mathbf{S}(u, v_{\max} - \Delta)$). However, by allowing normal deviation, the 4D-DP sampled space subsumes the space of samples covered by sampling between $\mathbf{S}(u, v_{\min}+\Delta)$ and $\mathbf{S}(u, v_{\max}-\Delta)$.

Our current implementation computes the optimal ruling approximation to periodic surfaces by using the line-surface metric. To employ the better SSHB, we need to overcome the (technical) difficulty of computing the composition across the boundary of $S(u, v)$. A possible solution can first split the u, v patch when it crosses the boundary, divide the split regions into rectangles and compute the composition for the divided pieces. In the future, we will work on how to incorporate the manufacturing constraints

Table 4
Computational Time (ms) on a Periodic Surface Fitting (5D-DP).

Boundary Sampling	Normal Elevation	Total Time (ms)
$100 \times 100 \times 100$	21×21	73,523
$50 \times 50 \times 50$	21×21	9,313
$50 \times 50 \times 50$	11×11	2,636

(e.g., collision between the cutter and the workpiece) into the procedure of optimal fitting.

Acknowledgment

This work was supported by the Hong Kong RGC/GRF Grant (CUHK/417508 and CUHK/417109) and the Direct Research Grant (CUHK/2050518). The research leading to these results has also received partial funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement PIAP-GA-2011-286426, and was supported in part by the Technion Vice President for Research Fund - Glasberg-Klein research fund.

References

[1] T.A. Spedding and Z. Wang, Study on modeling of wire EDM process, *Journal of Materials Processing Technology*, vol.69, pp.18-28. (1997)

[2] G. Elber and R. Fish, 5-axis freeform surface milling using piecewise ruled surface approximation, *Journal of Materials Processing Technology*, vol.119, pp.383-387. (1997)

[3] X.M. Zhang, L.M. Zhu, G. Zheng, and H. Ding, Tool path optimisation for flank milling ruled surface based on the distance function, *International Journal of Production Research*, vol.48, pp.4233-4251. (2010)

[4] H.-T. Hsieh and C.-H. Chu, Particle swarm optimisation (PSO)-based tool path planning for 5-axis flank milling accelerated by graphics processing unit (GPU). International Journal of Computer Integrated Manufacturing, vol.24, no.7, pp.676-687. (2011)

[5] S. Flöry and H. Pottmann, Ruled surfaces for rationalization and design in architecture, *Processings of Association for Computer Aided Design in Architecture (ACADIA)*, pp.103-109. (2010)

[6] H. Pottmann, Private communication.

[7] M.P. DoCarmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall. (1976)

[8] Z. Han, D.C.H. Yang, and J.J. Chuang, Isophote-based ruled surface approximation of free-form surfaces and its application in NC machining, *International Journal of Production Research*, vol.39, pp.1911-1930. (2001)

[9] J. Hoschek and U. Schwanecke, Interpolation and approximation with ruled surfaces, *The Mathematics of Surfaces VIII*, pp.213-231. (1998)

[10] X.F. Zha, A new approach to generation of ruled surfaces and its applications in engineering, *International Journal of Advanced manufacturing Technology*, vol.13, pp.155-163. (1997)

[11] T. Randrup, Approximation of surfaces by cylinders, *Computer-Aided Design*, vol.30, no.10, pp.807-812. (1998)

[12] Y. Zhou, J. Schulze, and S. Schäffler, Blade geometry design with kinematic ruled surface approximation, *Proceedings of the 2010 ACM Symposium on Applied Computing*, pp.1266-1267. (2010)

[13] C.C.L. Wang and K. Tang, Optimal boundary triangulations of an interpolating ruled surface, *ASME Journal of Computing and Information Science in Engineering*, vol.5, no.4, pp.291-301. (2005)
[14] H.-Y. Chen, I.-K. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner, On surface approximation using developable surfaces, *Graphical Models and Image Processing*, vol.61, no.2, pp.110-124. (1999)
[15] G. Elber, Model fabrication using surface layout projection, *Computer-Aided Design*, vol.27, pp.283-291. (1995)
[16] S. Flöry, Y. Nagai, F. Isvoranu, H. Pottmann, and J. Wallner. Ruled free forms. In L. Hesselgren et al., editors, *Advances in Architectural Geometry 2012*, pp.57-66, Springer. (2013)
[17] W. Wang, H. Pottmann, and Y. Liu, Fitting B-spline curves to point clouds by curvature-based squared distance minimization, *ACM Trans. Graph.*, vol.25, no.2, pp.214-238. (2006)
[18] F. Massarwi, C. Gotsman, and G. Elber, Paper-craft from 3D polygonal models using generalized cylinders, *Computer Aided Geometric Design*, vol.25, pp.576-591. (2008)
[19] J. Mitani, and H. Suzuki, Making papercraft toys from meshes using strip-based approximate unfolding, *Proceedings of ACM SIGGRAPH 2004*, pp.259-263. (2004)
[20] J. Subag and G. Elber, Piecewise developable surface approximation of general NURBS surfaces with global error bounds, *Proceedings of GMP'06*, pp.143-156. (2006)
[21] C.C.L. Wang, Towards flattenable mesh surfaces, *Computer-Aided Design*, vol.40, no.1, pp.109-122. (2008)
[22] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang, Geometric modeling with conical meshes and developable surfaces, *ACM Transactions on Graphics*, vol.27, no.3, pp.681-689. (2006)
[23] M. Kilian, S. Flöry, Z. Chen, N.J. Mitra, A. Sheffer, H. Pottmann, Curved folding, *ACM Transactions on Graphics*, vol.29, no.3. (2008)
[24] C.C.L. Wang, Computing length-preserved free boundary for quasi-developable mesh segmentation, *IEEE Transactions on Visualization and Computer Graphics*, vol.14, no.1, pp.25-36. (2008)
[25] G. Elber, and T. Grandine, Hausdorff and minimal distances between parametric freeforms in $\Re^2$ and $\Re^3$, *Proceedings of the 5th international conference on Advances in geometric modeling and processing (GMP'08)*, pp.191-204, Springer-Verlag. (2008)
[26] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, The MIT Press. (2009)
[27] IRIT, The irit geometric modeling environment 2010, `http://www.cs.technion.ac.il/irit`.
[28] T.-T. Cao, K. Tang, A. Mohamed, and T.-S. Tan, Parallel banding algorithm to compute exact distance transform with the GPU, *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pp.83-90. (2010)
[29] N. Stolte, Robust voxelization of surfaces, *Technial Report of Center for Visual Computing and Computer Science Department*, State University of New York at Stony Brook. (1997)
[30] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, Morgen-Kaufmann. (2002)
[31] S. Xiao, A.M. Aji, and W.-C. Feng, On the robust mapping of dynamic programming onto a Graphics Processing Unit, *15th International Conference on Parallel and Distributed Systems* (ICPADS), pp.26-33. (2009)
[32] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, *Computer Graphics Forum*, vol.17, no.2, pp.167-174. (1998)

Appendix: Surface-Surface Composition of Two Bézier Surfaces

Consider the 3-space Bézier surface

$$\mathbf{S}_1(u,v) = (x_1(u,v), y_1(u,v), z_1(u,v))$$
$$= \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{P}_{ij}\theta_i^n(u)\theta_j^m(v),\ \mathbf{P}_{ij} \in I\!R^3,\ u,v \in [0,1],$$

where $\theta_i^n(u) = \binom{n}{i}u^i(1-u)^{n-i}$ is the i'th Bézier basis function of order n and consider the planar Bézier surface

$$\mathbf{s}_2(r,t) = (u_2(r,t), v_2(r,t))$$
$$= \sum_{k=0}^{a}\sum_{l=0}^{b}(q_{kl}^u, q_{kl}^v)\theta_k^a(r)\theta_l^b(t),\ q_{kl}^u, q_{kl}^v \in I\!R,\ r,t \in [0,1].$$

The composition of $\mathbf{S}_{12}(r,t) = \mathbf{S}_1(\mathbf{s}_2(r,t))$ equals to

$$\begin{aligned}
&\mathbf{S}_{12}(r,t)\\
&= \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{P}_{ij}\theta_i^n(u_2(r,t))\theta_j^m(v_2(r,t))\\
&= \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{P}_{ij}\binom{n}{i}u_2(r,t)^i(1-u_2(r,t))^{n-i}\\
&\qquad\qquad \binom{m}{j}v_2(r,t)^i(1-v_2(r,t))^{m-j}\\
&= \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{P}_{ij}\binom{n}{i}\left(\sum_{k=0}^{a}\sum_{l=0}^{b} q_{kl}^u\theta_k^a(r)\theta_l^b(t)\right)^i\\
&\qquad\qquad \left(1-\sum_{k=0}^{a}\sum_{l=0}^{b} q_{kl}^u\theta_k^a(r)\theta_l^b(t)\right)^{n-i}\\
&\qquad\qquad \binom{m}{j}\left(\sum_{k=0}^{a}\sum_{l=0}^{b} q_{kl}^v\theta_k^a(r)\theta_l^b(t)\right)^i\\
&\qquad\qquad \left(1-\sum_{k=0}^{a}\sum_{l=0}^{b} q_{kl}^v\theta_k^a(r)\theta_l^b(t)\right)^{m-j}. \qquad (21)
\end{aligned}$$

Products and summations of polynomial Bézier basis functions are polynomials Bézier basis functions. Eq. (21) consists of merely summations and products of polynomial Bézier basis functions and hence is computable and representable as a Bézier surface (of high degrees).