Carving Shapes with Ruled Surfaces for Rough Machining

Paper ID: 245

Abstract

We propose a novel method for constructing a small set of ruled surfaces, each strictly outside the input shape, for rough machining. To generate such ruled surfaces, our algorithm consists of two phases: (1) generate a small set of covers for the input shape and (2) fit each cover patch using ruled surfaces without colliding with the input shape. Since the normal changes fastest along one of the principal curvature directions and the ruled surfaces cannot be deformed along their rulings, making the rulings and the principal curvature direction where the normal changes fastest orthogonal is more likely to increase the cover areas when using ruled surfaces to fit the surfaces. Meanwhile, we can trace another principal curvature direction to form the base curve. Accordingly, our cover generation step is driven by a smooth cross field approximating the principal curvature direction field. Central to the second stage is an optimization-based fitting to adaptively reduce the approximation error while keeping the ruled surfaces collision-free with the input shape. Our method's feasibility and practicability are demonstrated through various examples, including three physical manufacturing models using hot wire cutting.

Keywords: ruled surfaces, hot-wire cutting

Figure 1: Left: Hot Wire. Right: Diamond Wire. Images from `https://specialpatterns.com.au/` and `https://www.mactechonsite.com/`, respectively.

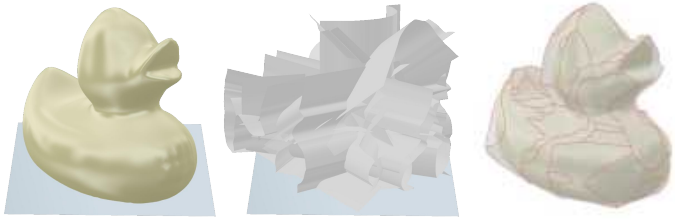


Figure 2: For an input shape with a fixed planar region (representing the workbench) (left), we generate a set of ruled surfaces (middle) which are collision-free with the input shape and fixed planar region so that cutting along these ruled surfaces removes as much material as possible (right).

1. Introduction

The subtractive fabrication process usually consists of two phases. One is rough machining, where the primary goal is to remove large amounts of material from a workpiece as quickly as possible [1, 2]. Another is using more precise techniques to shape the material for achieving the desired contours and dimensions necessary for the final product [3, 4]. Although rough machining is typically less precise, it is crucial for preparing the material for subsequent finer operations.

There are various ways of rough machining. One common method is to use regular wire to cut material, such as hot wire or diamond wire (Fig. 1). However, due to the limited motion space of robotic arms in diamond cutting and other methods, there is almost no optimization space. Therefore, we design our cutting path generation method for cutting methods with higher degrees of freedom, such as hot wire cutting. In this case, the surfaces formed by the movement of the regular wire are ruled surfaces. Given an input shape with a fixed planar region (representing the workbench), these ruled surfaces should satisfy the following requirements. First, each ruled surface is outside the input shape without any collision to ensure that the input shape is not destroyed after machining. Second, cutting along the constructed ruled surfaces leads to as much material removal as possible. Third, the number of ruled surfaces is small enough to improve the machining efficiency (Fig. 2).

In practice, users manually design such ruled surfaces; thus, it is a trial-and-error and time-consuming process. Our goal is to perform the design task automatically. However, this problem is rather challenging as computing ruled surfaces involves determining their numbers and placements under the overlap-free constraint. Generally, more material can be removed by introducing more ruled surfaces. Hence, the challenge is to find a favorable configuration (i.e., numbers and placements) of ruled surfaces.

Although [5] achieves globally collision-free ruled surfaces, their fitting error is high, and their method does not apply to complex models. Additionally, [6] explores hot wire cutting using curved lines; however, their method fails to guarantee collision-free and may result in overcutting. A potential approach for avoiding collisions is to scale up the input shape as the new input for [6]. However, finding an appropriate scale ratio requires manual tuning, which is tedious and may not generate results with low approximation error. Thus, it is unsuitable for practical machining processes. In general, the industry urgently needs an automatic algorithm that generates a small number of ruled surfaces to sufficiently approach the given shape without collisions.

In this paper, we propose a novel method to construct the desired ruled surfaces. We assume that the ruled surfaces are large enough to prevent the cutting tool from colliding with the material during the fabrication process. Our method is a two-step

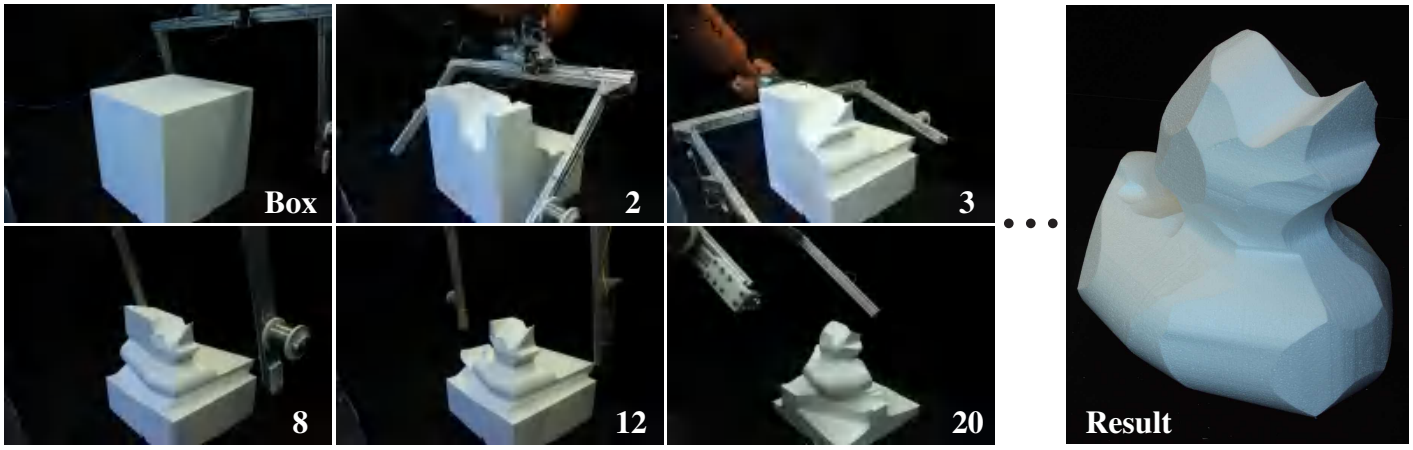


Figure 3: Given a shape fixed on the workbench, our algorithm automatically generates a small set of ruled surfaces that are collision-free with the input shape for rough machining. We achieve a trade-off between the number of ruled surfaces and the approximation error. Cutting along the optimized 23 ruled surfaces results in the rough machining result for the duck model. The number in the frame indicates the number of ruled surfaces that have been cut.

strategy that generates a small set of covers for the input and then fits cover patches using ruled surfaces without colliding with the input shape. However, developing a strategy to satisfy the requirements is challenging, and the reasons are twofold. First, the surface cover should facilitate the subsequent ruled surface fitting process, but this task has no well-known criterion. Second, ruled surface fitting with the collision-free constraint can be solved via a non-convex and nonlinear constrained optimization problem. However, it is often difficult to solve because the requirement that the ruled surface is sufficiently close to the input shape contradicts the collision-free constraint.

Our surface cover process is based on the following facts to overcome the first challenge. At a surface point, the normal change along one of the principal curvature directions is the fastest. Since the ruled surface can not be deformed along the rulings, setting the rulings and the principal curvature direction where the normal changes fastest orthogonal may increase the area that can be approximated well, called the cover area, thereby reducing the number of ruled surfaces. In this case, the other principal curvature direction can be used to guide the base curve directions. Consequently, we compute a smooth cross field close to the principal curvature direction field for surface cover to generate a set of cover patches. For the second challenge, we use a non-uniform B-spline surface (NUBS surface) to represent the ruled surface for fitting and present an interior point method to guarantee that the hard constraint is always met. To further reduce the approximation error, we adaptively add control points at high error areas and assign them different step sizes. Moreover, using different step sizes allows us to abandon the barrier function, which speeds up the fitting process and reduces the approximation error.

In summary, our technical contributions are as follows:

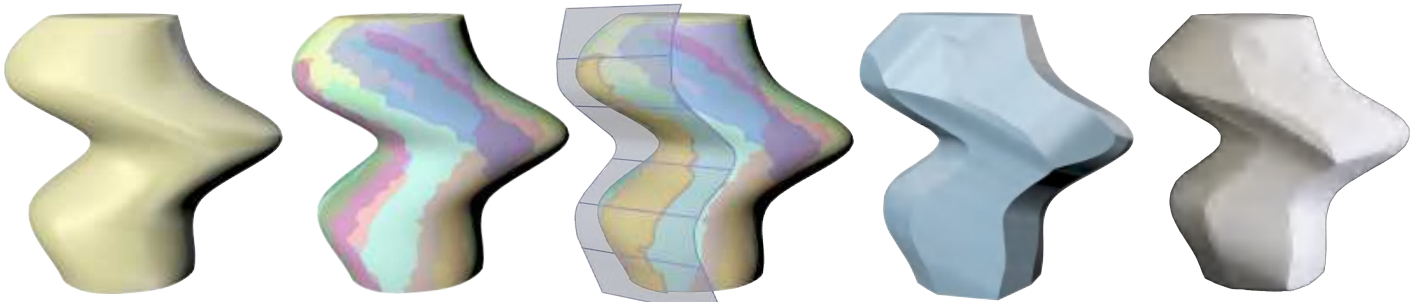
- proposing an automatic algorithm to generate collision-free ruled surfaces for rough machining.
- offering an efficient and effective fitting algorithm to make the resulting ruled surfaces approach the input shape without collisions.

To the best of our knowledge, our system is the first to generate a set of ruled surfaces for rough machining automatically. It can be directly applied to digital fabrication, providing significant practical value to this field as the rough machining problem has been a long-standing and unresolved issue in the industry. We apply the developed tool to various shapes to demonstrate the feasibility and effectiveness of our technique. Moreover, we fabricate three models physically using hot wire cutting along the generated ruled surfaces to show the system's usability and practicability.

2. Related Work

Rough machining. In subtractive manufacturing, the majority of rough machining work is based on a flat-bottomed cutter [7], which carves the external volume between the initial stock and the desired part shapes, following a predetermined tool path. As a result, rough machining typically accounts for 50% to 90% of the total machining time when a flat-bottomed cutter is used [8, 9]. A variety of strategies have been developed to tackle the challenge of planning roughing tool paths. These include the traditional slicing-and-filling technique [10], roughing paths derived from tessellated models [11], specialized approaches for impeller blades [12, 3], a method of volume decomposition for 3-axis CNC rough machining of freeform 3D shapes [13], and a vector field based volume peeling for multi-axis machining [14]. Unlike the previous methods, we focus on rough machining using ruled surfaces, employing a straight line cutter as the machining tool. In this area of research, [6] represents the most closely related work, where a deformed hot wire is applied to cut materials. However, their method may cut into the input model, violating the collision-free constraint, so it is unsuitable for rough machining. To our knowledge, this is the inaugural study to utilize ruled surfaces to conform to the desired part shape during the rough machining phase.

Ruled surface-driven machining. Beyond theoretical studies, many works have applied ruled surface fitting to practical machining processes. [15] use the trajectory of a cylindrical tool to approximate ruled surfaces, conceptualizing ruled surfaces as curves on Plücker's quadric and determining the tool axis position using the striction line of the ruled surface. [16] also use cylindrical tools to fit ruled surfaces, employing an optimization approach to calculate the position of the tool head by the error between it and three guiding curves on the ruled surface, although their method lacks an automated initialization of guiding curves. Besides cylindrical tools, [17] approximate ruled surfaces using the motion of a conical tool, proposing an optimization method by sequentially optimizing the tool axis position via guiding rail



(a) (b) (c) (d) (e)

Figure 4: The workflow of our algorithm. Given an input shape (a), our algorithm first generates a set of patches (shaded in different colors) to cover the input shape (b). Then, we apply an adaptive fitting process to fit these patches with ruled surfaces (rendered as translucent surfaces) (c). We show the simulation result after cutting along the generated ruled surfaces (d) and the physically fabricated result with hot wire cutting (e).

movements. Recently, significant research has emerged on hot wire cutting. [18] develop a feed rate adjustment rule. It automatically controls the tool motion at feedrate-sensitive corners based on a bisection method, thus limiting the maximum machining errors and improving the machining accuracy of the tool path in the five-axis flank milling of ruled surfaces. [19] use hot wire cutting to manufacture minimal surfaces, with the cutting path moving along the principal curvature lines of the surface, cutting material from both sides. [5] propose a conservative algorithm for finding collision-free tangential cutting directions in hot wire cutting, based on which simple model cutting paths can be proposed.

Ruled surface fitting. Linear tools are often used in the fabrication process to cut materials for subsequent processing. The surface generated by machining using such tools is the ruled surface. So, there has been considerable work in the fields of architecture and manufacturing on fitting freeform surfaces with ruled surfaces. [20] adopt a subdivision-based method to approximate general freeform surfaces with ruled surfaces. They continuously subdivide the input surface along parametric directions until the user-defined error threshold is met. However, their method requires extensive subdivision steps if high accuracy is demanded. [21] propose a rationalization method for complex surfaces commonly found in architecture, using ruled surface strips to approximate freeform surfaces. Their approach involves optimizing the strips to ensure curvature continuity at the boundaries, providing a theoretical framework for ruled surface fitting. Based on the fact that any negatively curved freeform shape can be approximated by a smooth union of ruled surface strips, [22] achieve surface approximation by following a given surface's asymptotic curves with conical ruled surfaces' rulings. Additionally, [23] presents an approach using dynamic programming to compute piecewise ruled surface approximations of freeform surfaces. Different from them, we focus on automatically generating the global collision-free piecewise ruled surfaces for approximating a given shape.

3. Method

3.1. Problem and formulation

Inputs and goals. The input is a shape represented as a triangular mesh $\mathcal{M}$ with a fixed planar region $\mathbf{P}_{\text{fix}}$ (representing the workbench). Without loss of generality, we assume that the input $\mathcal{M}$ is inside a box $\mathcal{B}$ with an edge length of 1, and the center of $\mathcal{M}$ is at the origin. We aim to automatically generate a set of ruled surfaces $\mathcal{R} = \{\mathbf{R}_i\}$, which satisfy the following requirements:

- *Collision-free requirement:* each ruled surface $\mathbf{R}_i$ does not collide with the input shape $\mathcal{M}$ and is outside $\mathcal{M}$.
- *Material removal requirement:* each ruled surface $\mathbf{R}_i$ is positioned as close to $\mathcal{M}$ as possible to remove more material volume of $\mathcal{B}$.
- *Efficiency requirement:* the number of the ruled surfaces is small to improve machining efficiency.

The first requirement is a hard constraint.

Methodology. It is challenging to achieve a favorable configuration (i.e., numbers and placements) of ruled surfaces as the requirements are complex and mutually restrictive. In practice, we observe that if the collision-free requirement is not considered and $\mathbf{R}_i$ is not required to be a ruled surface, constructing $\mathcal{R}$ satisfying other requirements is just a shape cover problem. Then, after covering $\mathcal{M}$ using a set of patches $\mathcal{P} = \{\mathbf{P}_i\}$, we can fit each patch $\mathbf{P}_i$ using ruled surfaces under the collision-free constraint.

Pipeline. According to the cover-to-fit idea, we develop a two-stage approach: (1) cover $\mathcal{M}$ with patches $\mathcal{P} = \{\mathbf{P}_i\}$ considering the property of the ruled surface, thereby facilitating the subsequent fitting stage (Section 4); and (2) robustly perform ruled surface fitting and adaptively add ruled surfaces to obtain a favorable tradeoff between the material removal requirement and the efficiency requirement while not intersecting $\mathcal{M}$ (Section 5). Fig. 4 shows the workflow of our method.

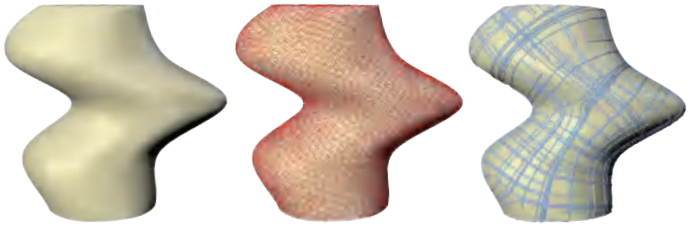
4. Surface cover

Ruled surfaces. A ruled surface is a surface swept out by a straight line moving along a curve. The straight lines themselves are called rulings. The ruled surface has a parameterization of the form:

$$\mathbf{s}(u, v) = \mathbf{c}(u) + v\mathbf{t}(u), \tag{1}$$

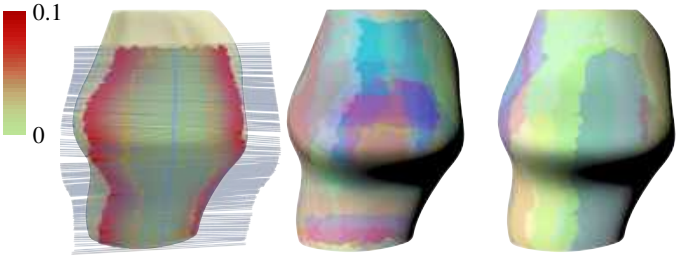
where $\mathbf{c}(u)$ is the base curve and $\mathbf{t}(u)$ is a unit vector representing the direction of the ruling.

Tracing principal directions. We aim to have each patch in the surface cover set as close as possible to a ruled surface to facilitate subsequent fitting. Therefore, we need a criterion to assess the proximity of a surface to a ruled surface. Our key idea is to construct an approximation of the closest ruled surface to the target patch, as the exact closest one is hard to get. Since a ruled surface cannot bend along its ruling lines, setting rulings and the direction of maximum normal variation of the target patch (i.e., principal direction) orthogonal may be beneficial for making ruled surfaces fit larger areas. So, we trace the principal curvature directions and construct simulated rulings orthogonal to them. These traced principal curvature direction lines are then used as the simulated base curve to generate the ruled surfaces.



(a) Input shape **(b)** Cross field **(c)** Direction lines

Figure 5: Given an input shape (a), we generate a smooth cross field (red crosses in (b)) that is used to find the direction lines L (blue lines in (c)).



(a) **(b)** **(c)**

Figure 6: (a) We construct many mimic rulings (the translucent lines) on a direction line (the blue line) to simulate a ruled surface. The triangles whose distance to these mimic rulings below d_{cover} are collected to form a patch (enclosed by the orange line). The color (encoded by the color bar) on the shape indicates the distance from the shape to the rulings. (b) The redundant patch set $\mathcal{P}_{\text{redundant}}$. (c) The final patch set $\mathcal{P}$.

Smooth cross fields. In practice, it is difficult to determine the principal curvature directions in relatively flat places, and the principal curvature direction field often contains many singularities. Thus, we compute a smooth cross field $\mathcal{F}$ to approximate the principal curvature direction field. We follow [24, 25] to generate such a cross field by fixing the principal directions. Specifically, the cross field is encoded as a complex polynomial, and the conjugacy constraints are constructed based on the principal curvature directions. A smooth cross field satisfying the geometric constraints is iteratively generated by harmonically interpolating the polynomial coefficients and minimizing a smoothness energy.

Coarse-to-fine strategy. After constructing a smooth cross field, we propose a coarse-to-fine strategy to generate the cover patches $\mathcal{P}$. The coarse step mimics the properties of the ruled surface to generate many patches (Section 4.1). Then, a patch selection is conducted to determine the final patches (Section 4.2).

4.1. The coarse step

Mimicking base curves. We trace the cross field $\mathcal{F}$ to form direction lines $\mathcal{L} = \{\mathbf{L}_i\}$ that mimic the base curve of ruled surfaces. We first randomly sample N_{seed} triangles as the starting points for tracing. To make the coarse step generate as many patches as possible for the subsequent refinement step, the two orthogonal directions given by the cross field $\mathcal{F}$ on each triangle are all used for tracing to obtain enough direction lines. Given one direction $\mathbf{d}$, we propagate forward along the directions $\mathbf{d}$ and $-\mathbf{d}$ starting from the center of $\mathbf{f}$ with the same speed (see Fig. 5). The propagation process terminates if any of the following conditions are satisfied: (1) the propagation front reaches the one-ring neighborhood of a singularity of $\mathcal{F}$; (2) the number of visited triangles exceeds $10\%N_{\text{tri}}$, where N_{tri} denotes the number of triangles of the input mesh. After propagation, we collect the ordered intersection points between the propagation rays and the mesh edges to form a direction line (denoted as $\mathbf{L} = \{\mathbf{p}_1, \cdots, \mathbf{p}_n\}$), which is then put into $\mathcal{L}$.

We limit the length of the direction lines as a termination condition because a too-long direction line may produce a patch with a complex shape, making subsequent ruled surface fitting difficult and causing a high approximation error. The too-short direction lines do not impact our final results as we abandon them in the fine step. If the input mesh is irregular, we can use the average edge length or the diagonal length of the bounding box for the termination condition.

Mimicking rulings. Along each direction line $\mathbf{L} = \{\mathbf{p}_1, \cdots, \mathbf{p}_n\}$, we construct many lines that mimic the rulings of the ruled surface. We select lines orthogonal to $\mathbf{L}$ and the triangle normals of the input shape to mimic the rulings. The triangle where two adjacent points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ locate is denoted as $\mathbf{f}_i$. We define a unit direction $\mathbf{r}_i = \mathbf{l}_i \times \mathbf{n}_i$, where $\mathbf{l}_i = (\mathbf{p}_{i+1} - \mathbf{p}_i)/\|\mathbf{p}_{i+1} - \mathbf{p}_i\|_2$ and $\mathbf{n}_i$ is the normal direction of $\mathbf{f}_i$. On $\mathbf{f}_i$, we generate $\lfloor \|\mathbf{p}_{i+1} - \mathbf{p}_i\|_2/\delta \rfloor + 1$ lines of length β in the direction $\mathbf{r}_i$ and centered at $\mathbf{p}_i^k$, where δ indicates a small distance and $\mathbf{p}_i^k = \mathbf{p}_i + k\delta\mathbf{l}_i, k = 0, \cdots, \lfloor \|\mathbf{p}_{i+1} - \mathbf{p}_i\|_2/\delta \rfloor)$, respectively. In our experiments, we set $\delta = 0.25 d_{\text{ae}}$ and $\beta = d_{\text{bb}}$, where d_{bb} denotes the length of the bounding box's diagonal and d_{ae} is the average edge length of the input shape.

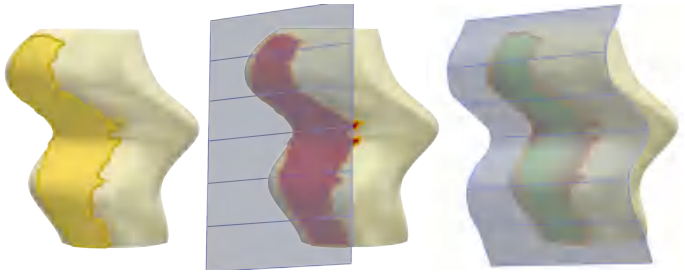
Patch construction. To simulate the following ruled surface fitting process under the global collision-free constraint, the generated line at point $\mathbf{p}_i^k$ is translated along the face normal by $d_{\text{translation}}$. The ruling lines without translations may intersect with the input model, thus violating collision-free constraints. As a result, the cover area may be reduced, increasing the number of patches. Therefore, we translate the ruling lines by a specific distance to balance the number of patches and the approximation error. It enlarges the cover area of each patch, thereby reducing the number of patches.

For each direction line $\mathbf{L}$, we first sample N_{sample} points on each generated line. Then, we perform the closest point projection for each sample point to the mesh, where the closest triangle is visited. If the minimum projection distance for a visited triangle is greater than d_{cover}, the triangle is marked as unvisited. Finally, we perform the flood fill algorithm from the triangles, where $\mathbf{L}$ passes through, to collect the visited triangles for forming a patch (Fig. 6 (a)). Since the mimic rulings represent the cutting tool, overcutting occurs once they collide with the input shape. So, we abandon the patch in which the mimic rulings intersect with the input shape as it violates the collision-free requirement. We run these steps for all direction lines to generate a redundant patch set $\mathcal{P}_{\text{redundant}}$ (Fig. 6 (b)). In our experiments, we set $N_{\text{sample}} = 200$, $d_{\text{translation}} = 1.5\%d_{\text{bb}}$, and $d_{\text{cover}} = 2.5\%d_{\text{bb}}$.

4.2. The fine step

Set cover. This step aims to select as few patches as possible from $\mathcal{P}_{\text{redundant}}$ to cover the input shape due to the efficiency requirement (see Fig. 6 (c)). We first perform a set cover algorithm [26] on $\mathcal{P}_{\text{redundant}}$ to achieve a minimum set cover $\mathcal{P}_{\text{set}}$.

Greedy refinement. The result of the set cover algorithm may include patches that highly overlap with others. To further reduce the number of patches, we develop a greedy algorithm to eliminate redundant patches. First, we sort the patches in $\mathcal{P}_{\text{set}}$ in descending order based on their area, denoted as $\{\mathbf{P}_{\text{sort,i}}\}$. The resulting patch set $\mathcal{P} = \{\mathbf{P}_{\text{sort,1}}\}$ is initialized by pushing the patch with the largest area into the set. Then, we iterate the sorted patches sequentially from a large to a small area. If a patch has 90% of its area covered by the patches in $\mathcal{P}$, we skip it; otherwise, we add it into $\mathcal{P}$. This process terminates until all the sorted patches are traversed or the union of the areas of all patches in $\mathcal{P}$ exceeds 99% of the area of



(a) (b) (c)

Figure 7: The fitting process. (a) The input shape $\mathcal{M}$ (the yellow model) and one patch $\mathbf{P} \in \mathcal{P}$ (the orange part) to be fitted. (b) The initial ruled surface. (c) The fitting result. The color on the patch in (b, c) indicates the distance from the patch point to the ruled surface with the color bar in Fig 6.

the input shape. For each triangle that is not covered, we assign it to its one adjacent patch in $\mathcal{P}$ if the patch has the smallest sum of the angle difference between its normal and the patch's average normal and the distance between its center and the patch center.

5. Ruled surface generation

5.1. Fitting algorithm

Problem overview. The inputs of the fitting process include the input shape $\mathcal{M}$ and one patch $\mathbf{P} \in \mathcal{P}$, which has a direction line $\mathbf{L}$ and mimicked rulings. We aim to generate a ruled surface $\mathbf{R}$ to approach $\mathbf{P}$ while not colliding with $\mathcal{M}$ (see Fig. 7).

Representation. We use a NUBS surface to represent the ruled surface $\mathbf{R}$. A B-spline curve in three-dimensional (3D) space is specified by a knot vector $\mathbf{u} = \{u_0, u_1, \ldots, u_m\}$ with $u_i \leq u_{i+1}$, basis functions of degree $p \in \mathbb{N}$ ($p = 3$ in our experiments), and control points $\{\mathbf{p}_i \in \mathbb{R}^3, i = 0, 1, \ldots, n\}$, with the relationship $m = p + n + 1$:

$$\mathbf{c}(u) = \sum_{i=0}^{n} B_{i,p}(u)\mathbf{p}_i, \ \forall u \in [u_0, u_m], \tag{2}$$

where the basis function $B_{i,p}(u)$ is:

$$B_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

$$B_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} B_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} B_{i+1,p-1}(u) \text{ if } p \geq 1. \tag{4}$$

Given two B-spline curves $\mathbf{c}_0, \mathbf{c}_1$ with identical knot vectors, the linear interpolation between corresponding points $\mathbf{c}_0(u), \mathbf{c}_1(u)$ sharing the same parameter u is performed to define the ruled surface $\mathbf{R}$, which has a parameterization of the form:

$$\begin{aligned} \mathbf{R}(u, v) &= (1 - v)\mathbf{c}_0(u) + v\mathbf{c}_1(u) \\ &= \sum_{i=0}^{n} B_{i,p}(u) \left\{(1 - v)\mathbf{p}_{i,0} + v\mathbf{p}_{i,1}\right\}, \end{aligned} \tag{5}$$

where $\mathbf{p}_{i,j} (j = 0 \text{ or } 1)$ is the control point of $\mathbf{c}_j$ and $v \in [0, 1]$. We collect the control points of the two B-spline curves into a set C and the knot vectors into a set $\mathcal{U}$.

Algorithm overview. The shape of the ruled surface is determined by these two sets C and $\mathcal{U}$. Accordingly, we apply an iterative strategy, which alternates in each iteration a pass to optimize the positions of control points and a pass to add control points (i.e., subdivide the knot vector) to generate the resulting ruled surface. First, our algorithm optimizes the position of control points to approach the input shape under the collision-free constraint (Section 5.1.1). The number of control points and the knot vector are fixed during this step. Then, we add new control points at the regions of the ruled surface with high approximation errors (Section 5.1.2). These two steps are alternately performed. Our initialization and termination conditions are shown in Section 5.1.3. Since adding new control points corresponds to subdividing the knot vector, it can be considered that we only optimize C. We denote the ruled surface determined by C as $\mathbf{R}_C$.

5.1.1. Updating positions of control points

Optimization problem. The fitting process is formulated as the following optimization problem:

$$\begin{aligned} \arg\min_{C} \quad & E(C) \\ \text{s.t.} \quad & \mathbf{R}_C \cap \mathcal{M} = \emptyset, \end{aligned} \tag{6}$$

where the optimization variable is C and $E(C)$ denotes the objective function.

Objective function. Our objective function contains two parts:

$$E(C) = E_{\text{error}}(C) + \omega E_{\text{smooth}}(C). \tag{7}$$

Here, ω ($\omega = 1$ in our experiments) is a weight to balance the approximation error and the smoothness energy. $E_{\text{error}}(C)$ is:

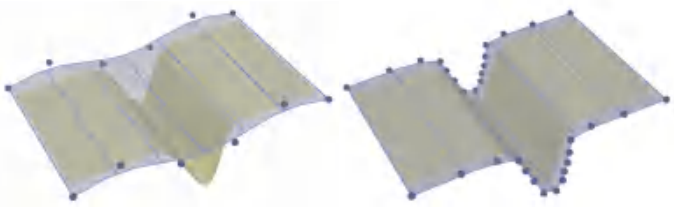
$$E_{\text{error}}(C) = \sum_{\mathbf{v}_k \in \mathbf{P}} \|\mathbf{v}_k - \mathbf{R}_C(u_k, v_k)\|_2^2, \tag{8}$$

where $\mathbf{v}_k$ denotes a vertex within $\mathbf{P}$ and $\mathbf{R}_C(u_k, v_k)$ represents the closest projection of $\mathbf{v}_k$ onto the ruled surface $\mathbf{R}_C(u, v)$. $E_{\text{smooth}}(C)$ is:

$$\begin{aligned} E_{\text{smooth}}(C) &= E_{\text{smooth,u}}(C) + \omega_{\text{uv}} E_{\text{smooth,v}}(C), \\ E_{\text{smooth,u}}(C) &= \sum_{i=0}^{n-2} \sum_{j=0}^{1} \|\mathbf{p}_{i,j} - 2\mathbf{p}_{i+1,j} + \mathbf{p}_{i+2,j}\|_2^2, \\ E_{\text{smooth,v}}(C) &= \sum_{i=0}^{n-1} \|(\mathbf{p}_{i,0} - \mathbf{p}_{i,1}) - (\mathbf{p}_{i+1,0} - \mathbf{p}_{i+1,1})\|_2^2, \end{aligned} \tag{9}$$

where the parameter ω_{uv} balances the smoothness terms along u-direction and v-direction. $E_{\text{smooth,u}}$ and $E_{\text{smooth,v}}$ control the smoothness of control points in the u-direction and v-direction, respectively. We will discuss other choices of $E_{\text{smooth,u}}$ in Section 5.1.3. ω_{uv} is set to 5 in the experiments.

Collision-free constraint. According to the convex hull property of the NUBS surface, we perform subdivisions to ensure that the ruled surface avoids collision with the input mesh $\mathcal{M}$. Specifically, subdividing the ruled surfaces in the guide curve direction k times and in the ruling direction l times yields sub-surfaces $\{\mathbf{R}_{i,j}\}$. If the convex hull of the control points of each $\mathbf{R}_{i,j}$ avoids collision with any mesh face $\mathbf{f} \in \mathcal{M}$, the ruled surface $\mathbf{R}$ does not intersect with the input shape $\mathcal{M}$. It is essential to select appropriate values for k and l to ensure that each subdivided subsurface $\mathbf{R}_{i,j}$ closely corresponds in scale to the faces of the input mesh. To enhance the precision of collision detection while optimizing the use of computational resources, we set k=8 and l=7 in our experiments.



Without adding control points Adaptively adding control points

Figure 8: Updating knot vectors to add control points.

Solver overview. We stick to the following principle to guarantee that the constraint is always met: initialize a constraint-satisfied ruled surface and then optimize it without violating the constraint. To optimize the problem (6) concerning two types of variables, i.e., C and the auxiliary variables $\mathbf{R}_C(u_k, v_k)$ in (8), we use the local-global solver. In the local step, we compute $\mathbf{R}_C(u_k, v_k)$ while fixing the control points C, i.e., fixing the ruled surface. The global step fixes (u_k, v_k) for $\mathbf{v}_k$ to update C.

Local step: updating the closest point. Given a vertex $\mathbf{v}_k$ within $\mathbf{P}$, we compute the closest point $\mathbf{R}_C(u_k, v_k)$ using an open-source toolkit called Open Cascade Technology [27]. Then, we establish a correspondence between the vertex $\mathbf{v}_k$ and the parameters (u_k, v_k). In the global step of updating the control points C, we assume that this correspondence is fixed.

Global step: updating positions of the control points. Given the correspondence between each $\mathbf{v}_k$ and (u_k, v_k), the objective function is quadratic concerning the positions of the control points C. We can easily minimize the quadratic function to obtain a solution $C^\star$ if the collision-free constraint is not considered. However, the ruled surface with $C^\star$ may intersect with $\mathcal{M}$.

A trivial solution for problem (6) is to use barrier functions to reformulate it as an unconstrained optimization problem. However, in our experiments, we observe that the barrier function not only prevents the ruled surface $\mathbf{R}_C$ from fitting the patch $\mathbf{P}$ well but also slows down the optimization process (see Fig. 15 (c) and more discussions in Section 6). So, we choose to abandon the barrier function and propose an adaptive line search to ensure our results meet the collision-free constraint. Specifically, we find that once there is a part of $\mathbf{R}_C$ close enough to $\mathbf{P}$, it may cause zero step size in the line search step, which prevents the approximation error from further reducing. To avoid this, we assign the same step sizes for a pair of control points $(\mathbf{p}_{i,0}, \mathbf{p}_{i,1})$, whereas adopting different step sizes for different pairs due to the locality of B-spline basis functions.

The descent direction for each control point is defined as the difference between the solved and the original positions. We update one pair of control points $(\mathbf{p}_{i,0}, \mathbf{p}_{i,1})$ while fixing other control points every time as follows:

1. We perform the Continuous Collision Detection (CCD) algorithm between the convex hull (generated by subdivision) and the input mesh to find the maximum allowable step size $t_{\max}$, ensuring that the collisions are avoided.
2. We perform a standard Armijo backtracking algorithm to determine the step size, which is initialized as $0.5t_{\max}$.

Each pair of control points corresponds to a segment of the ruled surface according to their influence range. The greater the E_{error} of the corresponding segment of a pair of control points, the earlier the pair of control points is updated.

5.1.2. Adding control points

The number of control points C has a significant impact on the fitting ability of the ruled surface $\mathbf{R}_C$. To further reduce the approximation error in the high-error areas, we increase the number of control points at the corresponding segments of the ruled surface $\mathbf{R}_C$ (Fig. 8). The new control points are added as follows. First, we sort the segments of the knot vector according to their approximation error. Then, we find the segment with as an large error as possible while satisfying the following condition to avoid generating too short segments through subdivision: the length of the ruled surface strip corresponding to this segment is greater than a threshold γ (we set $\gamma = 5d_{\text{ae}}$). More details about how to add control points are shown in the supplementary material. This process increases the number of control points, thereby reducing each control point's influence region to better accommodate complex shapes.

5.1.3. Other details

Initialization. Since the interior-point method is employed, each step produces a collision-free ruled surface, making a collision-free initialization essential. We choose a plane as the initial ruled surface with the following steps: (1) a B-spline curve with 6 control points is fitted to the given principal curvature lines; (2) a ruled surface is reconstructed from the fitted B-spline curve by optimizing the ruling to be orthogonal to the B-spline curve and the input shape's face normal; (3) the principal component analysis (PCA) method is used to reduce the dimensionality of the reconstructed ruled surface to a plane; (4) we translate the plane along its normal direction to a collision-free position as the initial plane. The details are shown in the supplementary material.

Termination conditions. The algorithm stops when the average distance E_{average}, i.e., the average value of Euclidean distances from vertices in $\mathbf{P}$ to the ruled surface, is less than a given threshold ϵ or when the number of iterations exceeds the specified maximum iteration number N_{iter}. In our experiments, we set $N_{\text{iter}} = 10$ and $\epsilon = 2\%d_{\text{bb}}$.

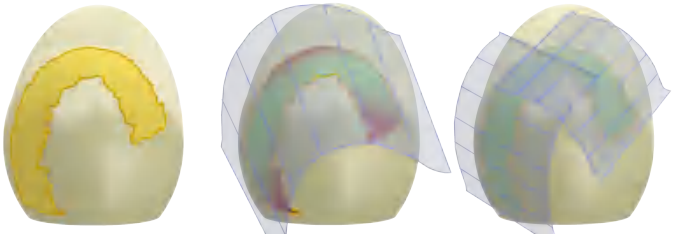
Testing multiple initializations. If $E_{\text{average}} > \epsilon$ after fitting for some patch, we rotate the initialization plane and run the fitting process to further reduce E_{average}. The rotation operation is uniformly performed N_{rotate} times both along the plane's normal and rulings. Then, we choose the result that has the smallest E_{average} as our fitting result. In our experiments, we set $N_{\text{rotate}} = 15$.

In practice, our initializations can produce fitting results that satisfy $E_{\text{average}} \leq \epsilon$ for most cover patches, eliminating the need for rotating sampling.

Smoothness term along u-direction. [28] provided a smoothness energy of a B-spline curve as follows:

$$E_{\text{smooth}}^{\text{curve}}(\mathbf{c}(u)) = \alpha \int_{u_0}^{u_m} \|c'(u)\|_2^2 du + (1-\alpha) \int_{u_0}^{u_m} \|c''(u)\|_2^2 du. \quad (10)$$

where $0 \leq \alpha \leq 1$ and $c(u)$ is the B-spline curve. This smoothness energy can also be used in (9) as the u-direction smoothness term. However, according to (4), the denominator of $c'(u)$ and $c''(u)$ contains the knots vector's interval length. Thus, subdividing the knot vector may cause the denominator to approach zero so that the value of $c'(u)$ and $c''(u)$ explodes. To avoid this issue, we finally choose to use $E_{\text{smooth, u}}$ in (9), a trivial smoothness term, to measure the smoothness of the ruled surfaces.



(a) input shape **(b)** before split **(c)** after split

Figure 9: We perform an adaptive split method to the patches have high approximation error. The color on the cover patch in (b, c) encodes the distance from the point of the cover patch to the ruled surface using the color bar in Fig 6.

5.2. Adaptive splitting and fitting

Motivation. After performing the above fitting process, we observe that E_{average} of some patches is still larger than ϵ (see Fig. 9 (b)). We think the reason is that the shape of these patches is too complex requiring the direction of the rulings to change drastically, so it is hard to use one ruled surface to fit it well. Thus, we introduce an adaptive splitting process to reduce the approximation error. Its core is to segment the patches with large errors into smaller patches for further fitting.

Segmentation. We use the elegant spectral mesh segmentation algorithm [29] to split a patch $\mathbf{P}$, given its fitting ruled surface $\mathbf{R}$. The segmentation algorithm is based on the following energy:

$$E_{\text{seg}} = \sum_{\mathbf{e}\in\mathcal{E}_I} E_{\text{rotation}}(\mathbf{e}) + \omega_{\text{seg}} E_{\text{distance}}(\mathbf{e}), \tag{11}$$

where $\mathcal{E}_I$ denotes the interior edge set of $\mathbf{P}$. E_{rotation} indicates the rotation difference between normals, E_{distance} denotes the distance error, and ω_{seg} is a positive parameter. E_{seg} is defined on the interior edges of $\mathbf{P}$, which is beneficial for segmentation.

We define E_{rotation} as follows:

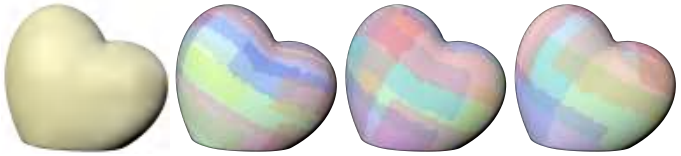
$$E_{\text{rotation}}(\mathbf{e}) = \|\mathbf{q}_{\text{patch}}(\mathbf{e}) - \mathbf{q}_{\text{ruled}}(\mathbf{e})\|_2^2. \tag{12}$$

Here, we denote the two adjacent faces of $\mathbf{e}$ as $\mathbf{f}_1$ and $\mathbf{f}_2$. The quaternion $\mathbf{q}_{\text{patch}}(\mathbf{e})$ represents the rotation between the two normal directions of triangles $\mathbf{f}_1$ and $\mathbf{f}_2$. We perform the closest point projection for the triangle centers of $\mathbf{f}_1$ and $\mathbf{f}_2$ to the ruled surface $\mathbf{R}$ to get two points. The rotation of the normal directions at these two points is denoted by $\mathbf{q}_{\text{ruled}}(\mathbf{e})$. E_{rotation} indicates the rotation difference between normals and estimates the second derivative of the approximation error, preventing the error from fluctuating greatly, which is crucial for fine machining. $E_{\text{distance}}(\mathbf{e})$ is the sum of the closest distance from the four vertices of triangles $\mathbf{f}_1$ and $\mathbf{f}_2$ to the ruled surface $\mathbf{R}$. It is four times E_{average} of the two neighbor triangles of $\mathbf{e}$, measuring the local approximation error.

The splitting process. We run the splitting process as follows:

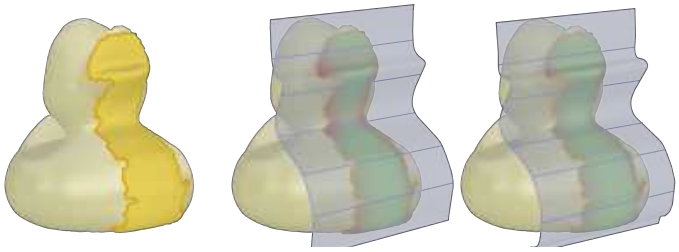
1. Select all the patches with E_{average} exceeding ϵ to form $\mathcal{A}$.
2. Pop one patch $\mathbf{P}$ from $\mathcal{A}$ and perform the segmentation on it to generate two sub-patches $\mathbf{P}_1$ and $\mathbf{P}_2$.
3. Use ruled surfaces to fit $\mathbf{P}_1$ and $\mathbf{P}_2$.
4. Add the sub-patch whose E_{average} exceeding ϵ to $\mathcal{A}$.
5. If $\mathcal{A}$ is empty, stop the process; otherwise, go to Step (2).

In our experiments, we perform a maximum of three split operations on each initial patch to avoid excessive patches. Actually, each initial patch is usually split only once. In practice, the connectivity of triangles can influence the energy defined on edges. In our experiments, the meshes are regular. For irregular meshes, subdividing the inputs is an effective strategy.



(a) Input shape **(b)** 2.0%d_{bb} **(c)** 2.5%d_{bb} **(d)** 3.0%d_{bb}

Figure 10: The cover parameter d_{cover}. From left to right, the number of cover patches progressively decreases, i.e., (21, 15, 10).



(a) Input shape **(b)** Uniform B-spline **(c)** Our used NUBS

Figure 11: Different representations of the ruled surfaces. The number of control points of the uniform B-spline is the same as our resulting NUBS surface. The color on the cover patch in (b, c) encodes the distance from the point of the cover patch to the ruled surface using the color bar in Fig 6.

6. Experiments and evaluations

We have tested our algorithm on various models to evaluate its performance, including three physical fabrication examples. Our algorithm was implemented in C++, and the experiments were executed on a desktop PC with a 3.80 GHz Intel Core i7-10700 and 32 GB of memory. Tables 1 and 2 summarize the statistics of our results. Unless otherwise specified, we use the color bar in Fig 6 to show the approximation error or distance.

6.1. Algorithm evaluations

The cover parameter d_{cover}. In the surface cover process, we use the parameter d_{cover} to control the area of each patch. We test different d_{cover}s to show its effect, as shown in Fig. 10. If d_{cover} is small (e.g., $d_{\text{cover}} = 2.0\%d_{\text{bb}}$ in Fig. 10 (b)), the area of each patch is small and the number of patches is great. Instead, a large d_{cover} (e.g., $d_{\text{cover}} = 3.0\%d_{\text{bb}}$ in Fig. 10 (d)) leads to a large area of each patch, making it challenging to fit each of them with low approximation errors. To achieve a favorable trade-off between the number of patches and the difficulty of the fitting process, we set $d_{\text{cover}} = 2.5\%d_{\text{bb}}$ in our experiments. We list the parameters of our algorithm in the appendix.

Representations of ruled surfaces. Uniform B-splines and NUBS can also be used to represent ruled surfaces. Compared to uniform B-splines, non-uniform B-splines (NUBS) are more effective in handling regions with high curvature variation. The key advantage lies in their non-uniform knot vectors. In uniform B-splines, knots are evenly spaced, which results in a fixed influence range for each control point. This fails to adaptively increase control point density in regions with rapid curvature variation, which lead to significant approximation errors (see Fig 11 (b)). In contrast, the knot vectors of NUBS can be adjusted according to curvature. This allows for more accurate approximation of complex geometries (see Fig 11 (c)). Hence, we use NUBS to represent ruled surfaces and apply an adaptive fitting method to handle this problem.

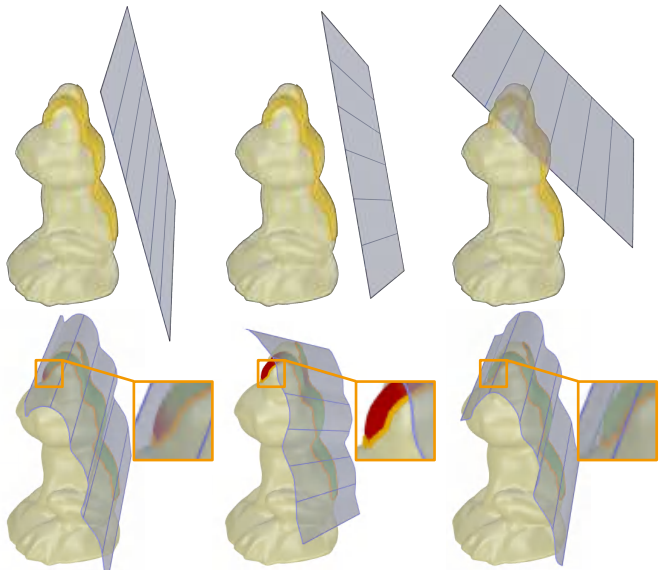
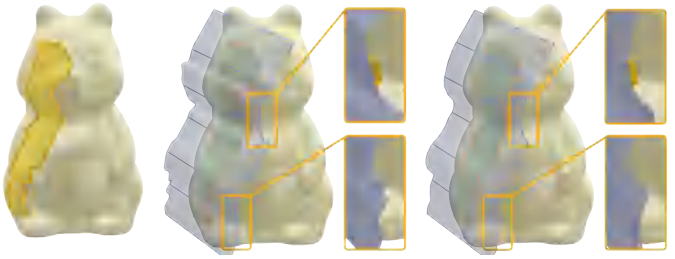


Figure 12: Different initializations for the fitting process. We show the initializations in the upper and the fitting results in the bottom.



(a) Input shape **(b)** Init step size = $t_{\max}$ **(c)** Init step size = $0.5t_{\max}$

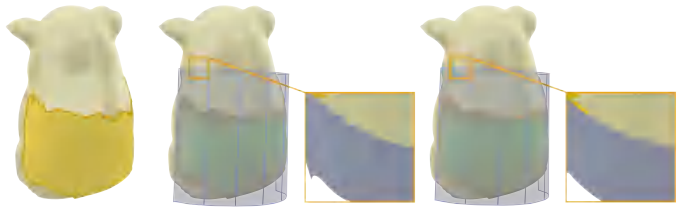
Figure 13: Different initial step sizes during the line search.

Initializations for the fitting process. The initial plane in the complex fitting process has a significant impact on the results. We test three initializations for the ruled surface fitting in Fig. 12. Thus, if the fitting error is large, we apply the rotational sampling to generate a suitable initialization (see the details in Section 5.1.3).

Parameters in fitting. There are two parameters in the fitting process that affect the shape of the final ruled surface. The first one is the initial step size in the line search for updating control points (Fig. 13). In our experiments, we set it $0.5t_{\max}$ where $t_{\max}$ is the maximum allowable step size satisfying the collision constraint. This is because decreasing the movement of control points in each iteration helps improve the fairing of the ruled surfaces. The second one is the minimum length of the ruled surface that can be subdivided in the updating knot vectors step. As shown in Fig. 14, if we do not limit the length of the ruled surface, it may be excessively subdivided, leading to ruled surfaces with fluctuation. (Fig. 14 (b)). Therefore, we set the minimum length to be d_{ae} in our experiments.

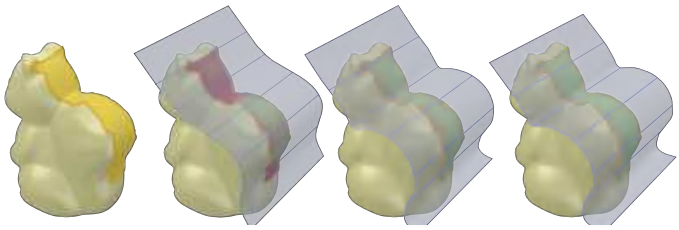
Other fitting solutions. [30] provide a trivial solution for solving the fitting problem. They employ a barrier function to ensure the ruled surface is always outside the input shape and utilize Newton's method to update the control points:

$$clog(x) = \begin{cases} -\frac{(x-x_0)^2}{x}\log\left(\frac{x}{x_0}\right) & \text{if } 0 < x \le x_0 \\ 0 & \text{elsewhere} \end{cases} \quad (13)$$



(a) Input shape **(b)** Without limitation **(c)** Minimum length = d_{ae}

Figure 14: Different minimum lengths for the ruled surfaces to be subdivided.



(a) Input shape **(b)** Barrier + uniform **(c)** Barrier + adaptive **(d)** Only adaptive

Figure 15: Different fitting strategies. (a) The input shape. (b) Using barrier functions and uniform step sizes (E_{average} = 0.00412, time = 9 minutes). (c) Using barrier function and different step sizes for control points (E_{average} = 0.00046, time = 30 minutes). (d) Only using different step sizes for control points (E_{average} = 0.00022, time = 6 minutes). The color on the cover patch in (b, c, d) encodes the distance from the point of the cover patch to the ruled surface using the color bar in Fig 6.

The variable x represents the distance from a sample point on the input model to the ruled surface and x_0 is a user-defined parameter. They then use this barrier function to reformulate the problem with hard constraints as an unconstrained one. It is easy to solve but still has problems. Using a uniform step size in the line search may lead to certain parts of the target surface not fitting well (see Fig. 15 (b)). This is because, to meet the no-collision constraint, areas that fit well contribute zero step sizes in the line search process. Since all control points share the same step size, the part of the ruled surface corresponding to the not well-fitted areas will stagnate, which results in great approximation error between the ruled surface and the target surface. To address this issue, we assign different step sizes to control points to achieve adaptive fitting with the barrier function (Fig. 15 (c)).

However, there are still two problems. First, the barrier function produces opposite descent directions in areas close to the target surface, trying to prevent the ruled surface from colliding with it. Second, computing different step sizes for each control point involves more computational complexity, which increases the method's running time.

In our experiments, we observe that if we abandon the barrier function and only use line search to ensure the two surfaces do not collide, we can quickly achieve a sufficiently low energy result (Fig. 15 (d)). Although the final results may not necessarily be local optimal solutions, nor can we achieve first-order optimality conditions since the descent direction we calculate is not the correct one, they are good enough for the fabrication process. Since the energy continuously decreases, we can set the step size as the termination condition. Almost all results satisfy the criterion of energy being below a threshold because the shape of the target patches is not complex. Since integrating strict collision-free constraints into the energy formulation is challenging and time-consuming, we use adaptive step sizes without the barrier function for efficient fitting with low approximation errors for practicality.

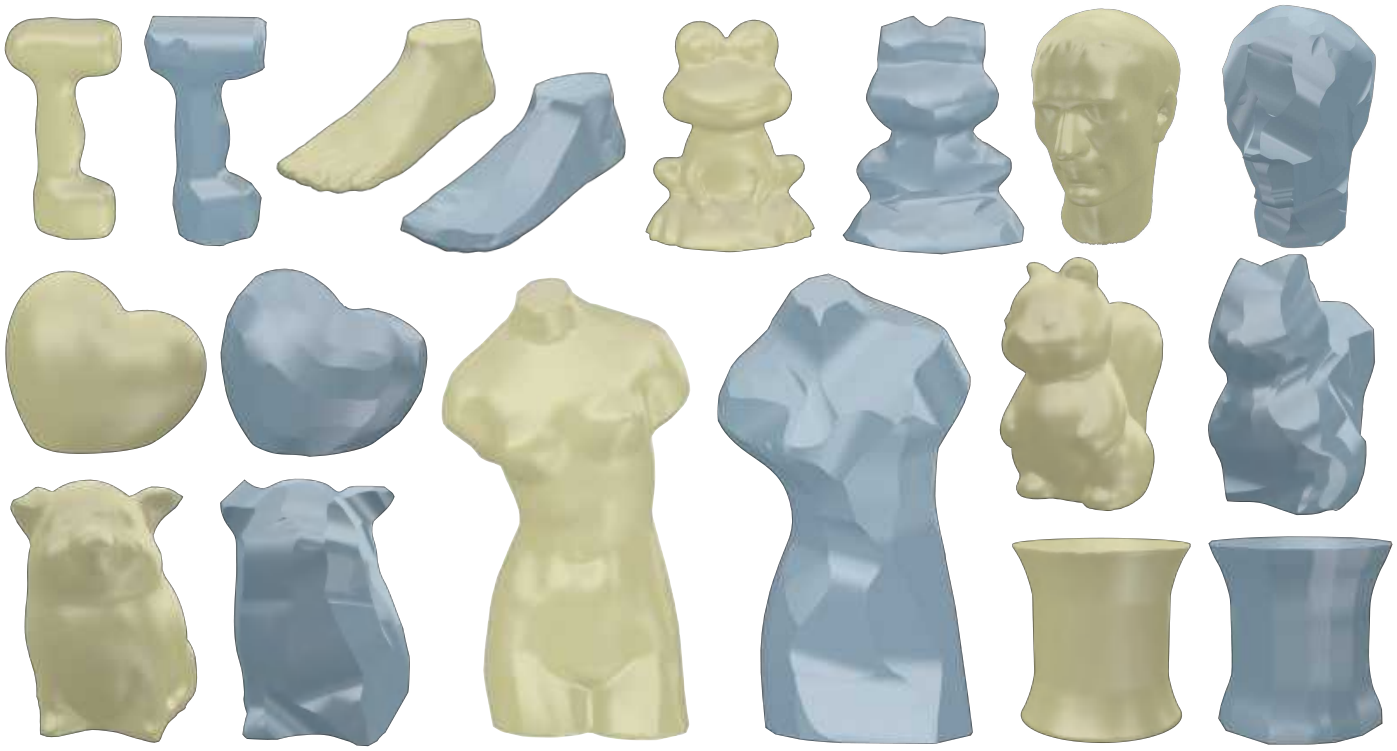
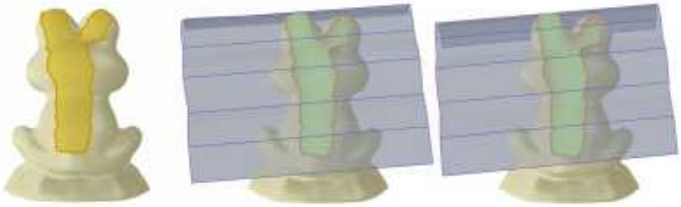


Figure 16: We succeed in generating a few ruled surfaces for rough machining for nine models.



(a) Input shape **(b)** Unidirectional error **(c)** Bidirectional error

Figure 17: Unidirectional error vs bidirectional error. (a) The input shape. (b) Unidirectional error (E_{average} = 0.00053, time = 9 minutes). (c) Bidirectional error (E_{average} = 0.00045, time = 15 minutes).

One-directional error vs bidirectional error. The E_{error} in Equ. 7 can also be measured by a bidirectional error. Fig. 17 shows the comparison between the bidirectional error result and ours. The experiment shows that the fitting error based on the one-directional distance is comparable to that of the bidirectional distance, but the computational efficiency is reduced when using the bidirectional error. To achieve a good trade-off between approximation error and computational efficiency, we use One-directional error as E_{error} in Equ. 7.

Smoothness term. The smoothness of a B-spline surface (e.g., its continuity C^k) is determined by its knot vectors and polynomial degree. However, this inherent smoothness only ensures local continuity in the parametric domain and does not directly guarantee geometric smoothness. When fitting complex shapes—particularly in regions with high curvature variation—relying solely on the parametric continuity of B-splines may lead to geometric artifacts such as folds or sharp transitions. To address this, additional smoothness terms (e.g., $E_{\text{smooth},u}$ and $E_{\text{smooth},v}$) are introduced to enforce global surface smoothness from a geometric perspective, rather than relying purely on parametric continuity (see Fig. 18 (b)). The smoothness term $E_{\text{smooth},v}$ is used to control the smoothness of the ruled surface along the v-direction. It is computed

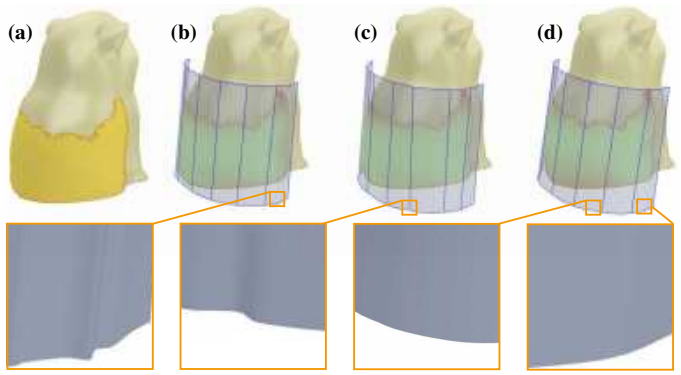
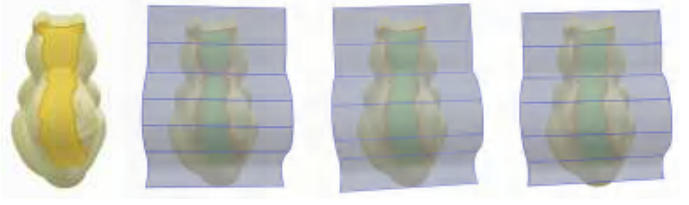


Figure 18: E_{smooth} term.(a) The input shape. (b) Only $E_{\text{smooth,u}}$ (E_{average} = 0.00071). (c) Only $E_{\text{smooth,v}}$ (E_{average} = 0.00062). (d) Ours (E_{average} = 0.00056).

based on the differences between corresponding control points along the v-direction. This helps suppress undesirable oscillations and promotes smoothness of the ruled surface (see Fig. 18 (c)).

The fitting error vs time. In our experiments, the number of control points can affect the fitting accuracy and efficiency. Fig. 19 shows the three results with different numbers of control points. For fairness, we control the number of final total control points by adjusting parameters ϵ and γ. As shown in Fig. 19, using fewer control points reduces computation time but results in higher fitting error. Conversely, using more control points yields accuracy comparable to our method, but results in a significantly higher computational cost. Based on this trade-off, we adopt $\epsilon = 3\%d_{\text{bb}}$ and $\gamma = 5d_{\text{ae}}$ as our default parameter settings.

Comparison with practitioner's results. We invite a practitioner from the field of industrial robot processing to generate the cutting paths with Rhinoceros 3D software [31]. We compare the



Input shape (20, 0.00057, 4) (32, 0.00022, 6) (68, 0.00018, 15)

Figure 19: The fitting error vs time. The parameter used in the second column are $\epsilon = 5\%d_{\text{bb}}$ and $\gamma = 6d_{\text{ae}}$. The third column shows the result produced using our method with its default parameter setting, which are $\epsilon = 3\%d_{\text{bb}}$ and $\gamma = 5d_{\text{ae}}$. The parameter used in the last column are $\epsilon = 1\%d_{\text{bb}}$ and $\gamma = 4d_{\text{ae}}$. The text below each result indicates the number of control points, E_{average}, and time consumption (mins), respectively.

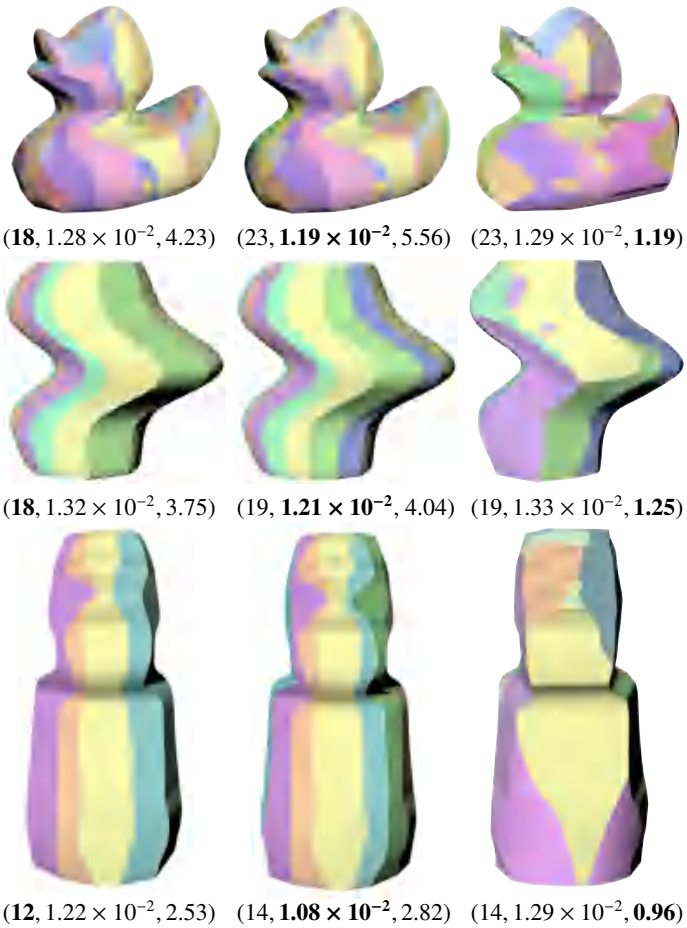
practitioner's results with three examples (see Fig. 20). To better evaluate our algorithm, the practitioner is asked to produce results under two termination conditions: (1) the E_{average} less than that of ours and (2) the number of cuts that are the same as ours. The experiment shows that the approximation error and the number of cuts of the practitioner's results are slightly better than that of ours in all three examples. However, the practitioner consumes much longer time than our method. Even in the example with the smallest gap (the first and last in the third row in Fig 20), the practitioner's time is 2.6 times ours (2.53 and 0.96 hours respectively). Our automated algorithm does not need manual labor while producing results with comparable quality to manual results. Thus, our method has significant practical value in digital fabrication.

More examples. We show nine models in Fig. 16. For these shapes, we successfully generate small sets of ruled surfaces for rough machining. Fig. 21 shows a simulation fabrication process, where cutting along the generated ruled surfaces leads to the rough machining result. The other three examples physically fabricated by the hot wire cutting are shown in Fig. 23. These simulation and fabrication results demonstrate the robustness and effectiveness of our algorithm.

Timings. The surface cover and the ruled surface fitting took about 1.32 minutes and 1.17 hours for the duck model with 6434 vertices and 23 ruled surfaces in Fig. 3. The fitting process occupies most of our method as we need to find proper step sizes for different control points so that the ruled surfaces satisfy the collision-free constraint and are close enough to the target patch. Among this process, the CCD for finding the maximum allowable step size is the most time-consuming. In addition, the collision-free initialization time in the ruled surface fitting is almost negligible. Although our algorithm is not fast enough, it is more time-saving compared to manual labor and is a fully automated algorithm. It produces results with low approximation error and a small number of cuts, thus having a significant practical value. We summarize the time consumption for all models in the paper in Table 1.

6.2. Physical fabrication

Hardware setup. Our physical setup includes a KUKA® Quantec KR120 R2700 Robot extra HA, which has a 6-degree-of-freedom arm with a working range of 2696mm (Fig. 22). In front of it, we have placed a workbench and a polystyrene foam cube with an edge length of 600mm, designated as the object to be cut. The hot wire cutting tool consists of a frame, flange, right-angle connectors, insulation blocks, springs, wire clamps, a coil of hot wire, and an adjustable DC power source. The frame is constructed by connecting one 6060 alloy profile and two 3060 aluminum profiles using right-angle connectors. The flange is mounted in the middle of the 6060 aluminum profile; two insulation blocks are mounted at the ends of the two 3060 aluminum profiles—one insulation block is equipped with a spring, and the other with a wire clamp. The hot wire coil is fixed on one of the 3060 aluminum profiles, and the hot wire is drawn from the coil. One end of the hot wire passes through the wire clamp and is fixed on the spring of the other 3060 aluminum profile. The wire clamp and



(**18**, 1.28×10^{-2}, 4.23) (23, $\mathbf{1.19 \times 10^{-2}}$, 5.56) (23, 1.29×10^{-2}, **1.19**)

(**18**, 1.32×10^{-2}, 3.75) (19, $\mathbf{1.21 \times 10^{-2}}$, 4.04) (19, 1.33×10^{-2}, **1.25**)

(**12**, 1.22×10^{-2}, 2.53) (14, $\mathbf{1.08 \times 10^{-2}}$, 2.82) (14, 1.29×10^{-2}, **0.96**)

Figure 20: Comparison with practitioner's results. The first column shows the practitioner's results which are produced when their E_{average} is less than that of ours. The second column includes the practitioner's results with the same number of cuts as ours. Our results are in the last column. The text below each result indicates the number of cuts, average error, and time consumption (hours), respectively.

Models	Number of cuts	Our result tol 2% d_{bb}	Our result tol 3% d_{bb}	$d^{\text{avg}}_{\text{result}}$	d_{bb}	t_{cover} (h)	t_{fitting} (h)
Fig. 3 Duck	23	94.88%	98.85%	1.29×10^{-2}	1.63	2.19×10^{-2}	1.17
Fig. 4 Artwork	19	94.34%	98.94%	1.33×10^{-2}	1.58	2.11×10^{-2}	1.13
Fig. 16 Drill	17	92.89%	98.68%	0.95×10^{-2}	1.06	2.08×10^{-2}	1.60
Fig. 16 Foot	22	96.58%	99.88%	1.01×10^{-2}	1.13	2.53×10^{-2}	1.92
Fig. 16 Frog	38	92.29%	97.84%	1.01×10^{-2}	1.43	4.64×10^{-2}	3.03
Fig. 16 Head	31	96.80%	99.33%	0.98×10^{-2}	1.51	3.47×10^{-2}	2.65
Fig. 16 Heart	25	99.76%	100.00%	0.93×10^{-2}	1.67	2.39×10^{-2}	0.92
Fig. 16 Pig	27	93.13%	98.57%	1.31×10^{-2}	1.47	2.92×10^{-2}	2.35
Fig. 16 Squirrel	42	93.28%	98.01%	1.29×10^{-2}	1.48	5.36×10^{-2}	3.40
Fig. 16 Venus	31	96.62%	99.66%	1.02×10^{-2}	1.21	2.61×10^{-2}	1.68
Fig. 16 Barrel	19	99.95%	100.00%	1.14×10^{-2}	1.73	1.56×10^{-2}	0.47
Fig. 21 Moai	14	87.75%	95.09%	1.29×10^{-2}	1.21	1.69×10^{-2}	0.98

Table 1: Overview of our simulation shapes. We measure the distance from our simulation shape to the input shape to define the error. $d^{\text{avg}}_{\text{result}}$ is average distance. The third (or fourth) column shows the ratio of the area within the specified error threshold to the total area of the simulation shape. t_{cover} and t_{fitting} denote the time consumed by the surface cover process and the fitting process, respectively. These times are measured in hours.

Figure 21: A simulation fabrication process of one example. The left three columns are the first, second, and fourteenth cuts. The last two columns show the input shape and the simulation shape generated by the simulated hot wire cutting.

Figure 22: Hardware Setup. Robotic arm product model: KUKA® Quantec KR120 R2700 extra HA Robot.

Models	Number of cuts	Our result tol ±2% d_{bb}	Our result tol ±3% d_{bb}	$d_{\text{fab}}^{\text{avg}}$	d_{bb}
Fig. 3 Duck	23	98.96%	99.80%	1.00×10^{-2}	1.63
Fig. 4 Artwork	19	100.00%	100.00%	0.49×10^{-2}	1.58
Fig. 21 Moai	14	92.26%	98.69%	0.85×10^{-2}	1.21

Table 2: Overview of the fabrication results. We use the one-sided distance from the fabrication shape to our simulation shape to measure the fabrication error. $d_{\text{fab}}^{\text{avg}}$ is mean error in the fabrication. The third (or fourth) column shows the ratio of the area within the specified error threshold to the total area of the fabrication surface.

spring are equipped with lead-out wires connected to the positive and negative terminals of the adjustable DC power source. In our experiments, we set the power source current to 2.0 amperes and the voltage to 20.0 volts, heating the hot wire to 250 degrees Celsius. The movement speed of a point on the hot wire is set to 2mm/s, ensuring that the foam melts before contacting the hot wire, thereby avoiding the impact of resistance.

Motion trajectory. We use NUBS to represent the motion trajectory of the robotic arm. The motivation is that smooth B-spline surfaces typically result in lower machining errors than polylines in practical manufacturing. This is because the robotic arm must pause or significantly decelerate at the turning points of polyline paths or accelerate abruptly when changing direction. Such actions can cause excessive dwelling at sharp corners, resulting in unwanted material melting or inducing deformation due to rapid directional changes, thus negatively affecting fabrication accuracy.

Machining order. We assume that the ruled surfaces representing the cutting surfaces are sufficiently wide. Hence, the machining order has almost no impact on the fabrication results. Besides, mature algorithms (e.g., KUKA—prc) have been developed in the industry to connect multiple cutting paths. Therefore, we study how to generate each cutting path rather than the complete path.

Besides, to ensure the model does not fall off the material in advance during the cutting process, we manually select the last cutting path in the software.

Entry and exit cut. In the physical fabrication process, we need to specify additional cutting paths for entry and exit cuts, which are not discussed in previous work [5, 6]. This is because the cutting tool (a hot wire in our experiments) has a high temperature, and if it is close to the surface of the input shape, the material may be burned and damaged. In our algorithm, since the ruled surface we used to fit the patch is large enough, the redundancy part of the ruled surface naturally provides a path for entry and exit cuts. We only need to set the hot wire cut along the ruled surfaces, and the material will be protected from being burned.

Fabrication constraints. Since the material is placed on a platform in front of the cutting machine, we need to consider whether the robotic arm collides with the platform and the material during the fabrication process. To ensure that our results meet this fabrication constraint, we add a bottom plane for the input shape. The method is as follows. For each input shape, we manually add a rectangular plane as the bottom plane at an appropriate position. Then, this bottom plane is added to the collision detection in our algorithm to generate a set of ruled surfaces that do not intersect with the input shape and the bottom plane. So, at this point, the cutting path generated based on our results will naturally not cause collisions between the robotic arm and the platform. The collisions between the robotic arm and the material can be eliminated by appropriately setting the size of the unprocessed material $\mathcal{B}$. When $\mathcal{B}$ is small enough, there exists a way to place the robotic arm to avoid collisions during cutting. In our experiments, we set the edge length of $\mathcal{B}$ as half the length of the hot wire.

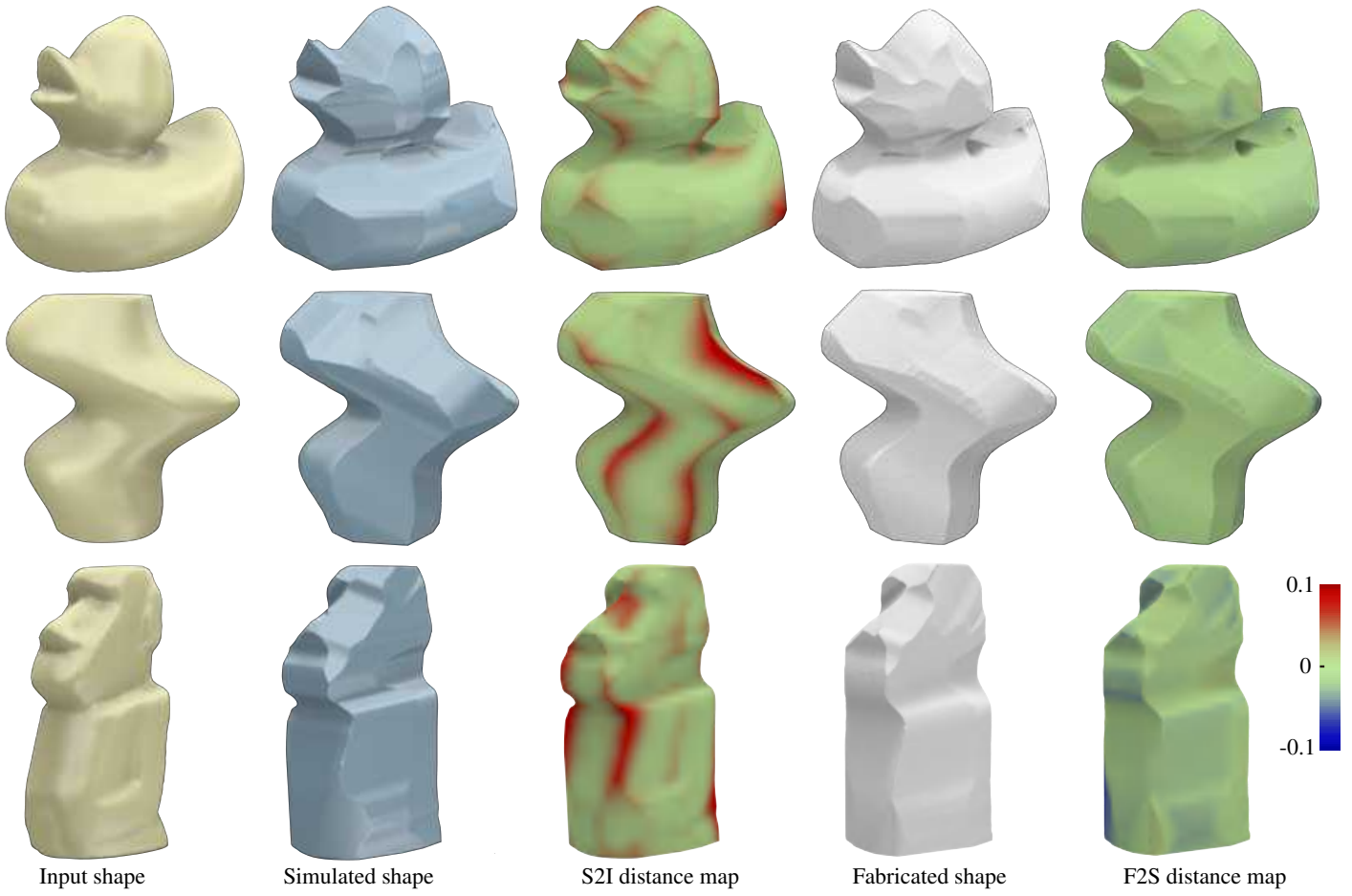


Figure 23: Simulation and fabrication errors. The term "S2I" refers to simulation shapes to input shapes, and "F2S" refers to fabrication shapes to simulation shapes. The distance map is color-coded by the color bar.

Analysis. We test our algorithm on 12 different shapes. Table. 1 shows the analysis of these results. For these complex shapes, our algorithm can achieve results with small approximation errors using about 30 cuts (see Table. 1 second column). Only the Frog and Squirrel models need to be cut by 38 and 42 times. The distances from our simulation shape to the input shape in most areas are below 2% d_{bb} and almost the distances of all parts are below 3% d_{bb} (Table. 1 third and fourth columns). We also evaluate the accuracy of the fabrication results (see Table. 2). We use a 3D scanner to reconstruct these models and then compute the distance from the fabrication results to our simulation results. The two errors are small and comparable. This shows our algorithm is enough to use in the physical fabrication process.

7. Conclusion and Discussions

We propose a novel method to compute a small set of ruled surfaces for rough machining. Each ruled surface is strictly outside the input shape. A cover-to-fitting strategy is developed to achieve a good trade-off between the number of ruled surfaces and the approximation error under the intersection-free constraints. Specifically, our adaptive fitting strategy makes ruled surfaces fully approximate the input surface without using a barrier function. We also fabricated three models using hot wire cutting to demonstrate the feasibility and practicability of our algorithm. To the best of our knowledge, our work is the first to offer an automated solution to this problem. Although our algorithm has a slow overall speed, we believe it has practical value due to its fully automated nature and sufficiently small approximation error of the results.

Choice of initial ruled surfaces. Constructing the initial ruled surfaces based on contour lines rather than using planar surfaces provides a more geometrically reasonable initial surface, which can guide the optimization away from a local optimum and accelerate convergence. However, contour lines can only generate limited types of ruled surfaces, such as cylindrical and planar surfaces, so they cannot handle conical surfaces well. How to solve this difficulty and apply contour lines to our algorithm is a worthwhile direction for future research.

Approximation error. For an arbitrary model, we can not ensure that the approximation error between the final result and the input shape is always less than a specified threshold. Since the purpose of rough machining is to efficiently remove as much material as possible, rather than obtaining a precise model, we only carry out a limited number of patch splits to reduce the approximation error.

Joint patch optimization. The patches are optimized independently rather than jointly, meaning that the partition boundaries cannot be adjusted during optimization. This restriction prevents the correction of large errors at patch intersections and limits the overall surface fitting quality. Although joint optimization can improve fitting results, it significantly increases the cost of fitting.

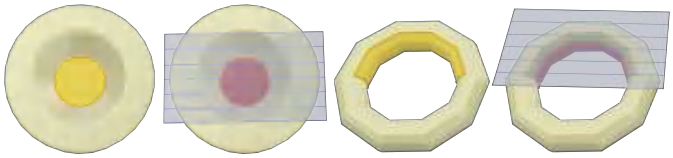


Figure 24: Left: an example of cutting concave shape. Right: an example of cutting high-genus shape.

This is primarily due to the need to simultaneously optimize thousands of control points while performing collision detection. The time required to solve such a large-scale optimization problem becomes practically unacceptable. Moreover, joint optimization would necessitate dynamically adjusting the fitting region. How to properly assign fitting regions to each ruled surface while ensuring full coverage of the input shape remains a challenging problem.

Concave shapes. Regular hot wire cutting machines cannot handle some kinds of concave shapes because the tool will collide with the input shape when cutting the concave part with a large depth. So, to maintain the collision-free constraint, the distance from this concave part to the ruled surface should be larger than the depth, which makes it unacceptable for rough machining (Fig. 24 left). This shortcoming is inherent in this kind of machining method. Using curves instead of straight lines as the hot wire can, to a certain extent, solve this problem [6]. It is worth exploring in this direction in the future.

None-zero genus shapes. Similar to concave shapes, there are also areas on none-zero genus shapes that can not be cut by the hot wire (Fig. 24 right). A feasible solution is to partition the input shape into several genus-zero parts and process them separately. However, it may compromise the integrity of our results. Another simple way is to manually make holes in the input shape and then reassemble the hot wire to cut through the holes. However, this method is undoubtedly very cumbersome. In the future, we aim to improve our algorithm to be able to deal with high-genus shapes.

The type of the input shape. Since our method utilizes the principal curvature direction field to guide the generation of the ruled surfaces, it is unsuitable for handling shapes with noisy surfaces. In this case, the principal curvature direction field is not smooth, which makes the lengths of the mimic base curves short. This decreases the area each path can cover and then increases the number of ruled surfaces in the final results. A feasible solution is to construct a cage for this input shape and run our method to process the cage. In addition, shapes with many planes, such as CAD models, are also not suitable for our method to handle. This is because the principal curvature directions of a plane can be arbitrary, while our method just traces one of them. Our method does not optimize the principal curvature direction to ensure no collision between the ruled surface constructed from the principal curvature direction and the input shape. Designing a dedicated algorithm is a better choice for these special cases, as our algorithm is more inclined to handle models with general shapes.

References

[1] Z. Yin, Rough and finish tool-path generation for nc machining of freeform surfaces based on a multiresolution method, Computer-Aided Design 36 (12) (2004) 1231–1239.

[2] G. Vickers, Z. Dong, H. Li, Optimal rough machining of sculptured parts on a cnc milling machine, Journal of Engineering for Industry, Transactions of the ASME 115 (4) (1993) 424–432.

[3] H.-T. Young, L.-C. Chuang, K. Gerschwiler, S. Kamps, A five-axis rough machining approach for a centrifugal impeller, The International Journal of Advanced Manufacturing Technology 23 (2004) 233–239.

[4] S. Tao, K.-L. Ting, Unified rough cutting tool path generation for sculptured surface machining, International Journal of Production Research 39 (13) (2001) 2973–2989.

[5] B. van Sosin, M. Bartoň, G. Elber, Accessibility for line-cutting in freeform surfaces, Comput. Aided Des. 114 (2019) 202–214.

[6] S. Duenser, R. Poranne, B. Thomaszewski, S. Coros, Robocut: Hot-wire cutting with robot-controlled flexible rods, ACM Trans. Graph. 39 (4) (2020) 98–1.

[7] Y. Hu, W. Tse, Y. Chen, Z. Zhou, Tool-path planning for rough machining of a cavity by layer-shape analysis, The International Journal of Advanced Manufacturing Technology 14 (5) (1998) 321–329.

[8] E.-Y. Heo, D.-W. Kim, B.-H. Kim, D.-K. Jang, F. F. Chen, Efficient rough-cut plan for machining an impeller with a 5-axis nc machine, International Journal of Computer Integrated Manufacturing 21 (8) (2008) 971–983.

[9] J. E. Petrzelka, Geometric process planning in rough machining (2009).

[10] M. Liang, S. Ahamed, B. Van Den Berg, A step based tool path generation system for rough machining of planar surfaces, Computers in Industry 32 (2) (1996) 219–231.

[11] M. Balasubramaniam, P. Laxmiprasad, S. Sarma, Z. Shaikh, Generating 5-axis nc roughing paths directly from a tessellated representation, Computer-Aided Design 32 (4) (2000) 261–277.

[12] Y. Chen, G. Mei, M. Lv, J. Gao, Tool path generation for efficient rough-cut machining of ruled surface impellers, in: 2014 International Conference on Mechatronics and Control (ICMC), 2014, pp. 599–602.

[13] A. Mahdavi-Amiri, F. Yu, H. Zhao, A. Schulz, H. Zhang, Vdac: volume decompose-and-carve for subtractive manufacturing, ACM Transactions on Graphics (TOG) 39 (6) (2020) 1–15.

[14] N. Dutta, T. Zhang, G. Fang, I. E. Yigit, C. C. Wang, Vector field based volume peeling for multi-axis machining, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Vol. 87295, American Society of Mechanical Engineers, 2023, p. V002T02A025.

[15] K. Sprott, B. Ravani, Cylindrical milling of ruled surfaces, The International Journal of Advanced Manufacturing Technology 38 (2008) 649–656.

[16] H. Gong, L.-X. Cao, J. Liu, Improved positioning of cylindrical cutter for flank milling ruled surfaces, Comput. Aided Des. 37 (12) (2005) 1205–1213.

[17] C. Li, S. Bedi, S. Mann, The International Journal of Advanced Manufacturing Technology 29 (2006) 1115–1124.

[18] C. Chu, W. Huang, Y. Hsu, Machining accuracy improvement in five-axis flank milling of ruled surfaces, International Journal of Machine Tools and Manufacture 48 (7-8) (2008) 914–921.

[19] H. Hua, T. Jia, Wire cut of double-sided minimal surfaces, The Visual Computer 34 (6) (2018) 985–995.

[20] L. Piegl, W. Tiller, The nurbs book, springer-verlag, New York (1995).

[21] S. Flöry, H. Pottmann, Ruled surfaces for rationalization and design in architecture, LIFE in: formation. On responsive information and variations in architecture (2010) 103–109.

[22] S. Flöry, Y. Nagai, F. Isvoranu, H. Pottmann, J. Wallner, Ruled free forms, in: Advances in architectural geometry 2012, Springer, 2013, pp. 57–66.

[23] C. C. Wang, G. Elber, Multi-dimensional dynamic programming in ruled surface fitting, Comput. Aided Des. 51 (2014) 39–49.

[24] O. Diamanti, A. Vaxman, D. Panozzo, O. Sorkine-Hornung, Designing n-polyvector fields with complex polynomials, Comput. Graph. Forum 33 (5) (2014) 1–11.

[25] D. Bommes, H. Zimmer, L. Kobbelt, Mixed-integer quadrangulation, ACM Trans. Graph. 28 (3) (2009).

[26] W. Lin, F. Ma, Z. Su, Q. Zhang, C. Li, Z. Lü, Weighting-based parallel local search for optimal camera placement and unicost set covering, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, 2020, pp. 3–4.

[27] O. C. S.A.S., Opencascade technology, `http://www.opencascade.com` (2020).

[28] Y. Liu, H. Yang, W. Wang, Reconstructing b-spline curves from point clouds–a tangential flow approach using least squares minimization, in: International Conference on Shape Modeling and Applications 2005 (SMI'05), IEEE, 2005, pp. 4–12.

[29] W. Tong, X. Yang, M. Pan, F. Chen, Spectral mesh segmentation via ℓ_0 gradient minimization, IEEE. T. Vis. Comput. Gr. 26 (4) (2018) 1807–1820.

[30] R. Ni, T. Schneider, D. Panozzo, Z. Pan, X. Gao, Robust & asymptotically locally optimal uav-trajectory generation based on spline subdivision, in: 2021 IEEE International Conference on Robotics and Automation (ICRA),

IEEE, 2021, pp. 7715–7721.
[31] R. McNeel, Associates, Rhinoceros 3d (2024).
URL https://www.rhino3d.com/

Appendix A. Adding control points details

We use the following criterion to add control points:

$$N_{\text{new}} = \begin{cases} 5 & \text{if } E_{\text{interval}}>3\epsilon \text{ and } L_{\text{interval}}>2\gamma \\ 0 & \text{if } E_{\text{interval}}<\epsilon \text{ or } L_{\text{interval}}<\gamma \\ 3 & \text{otherwise} \end{cases}, \tag{A.1}$$

where N_{new} is the number of new points, E_{interval} is this interval's approximation error E_{error} and L_{interval} is this interval's length. The positions of the new points are determined by dividing the interval into $N_{\text{point}} + 1$ equal segments, with the maximum value of N_{point} limited to 5.

Appendix B. Initialization details

Step 1: The B-spline $\mathbf{c}(u)$ fitting problem can be formulated as follows:

$$\arg\min_{C} E_{\text{dist}} = \sum_{k=0}^{K} \|\mathbf{v}_k - \mathbf{c}(u_k)\|_2^2 / |\mathcal{V}|, \tag{B.1}$$

where $\mathcal{V} = \{\mathbf{v}_k\}$ are the sampling points on the fitted principal curvature line $\mathbf{L}$, $\mathbf{c}(u_k)$ is the closest point to v_k and C is the control points of $\mathbf{c}(u)$. The initialization of C is set to some samples in $\mathcal{V}$. The first and last control points are constrained to be the start and end points of $\mathbf{L}$.

This problem is easy to solve because it is a least square problem by solving the normal equations. We also adaptively add control points to make the fitting result better. The iteration stops when E_{dist} is less than α. In our experiments, we set α to the average edge length of the input mesh.

Step 2: We uniformly sample the knot vector of $\mathbf{c}(u)$ to obtain $\{u_m\}$. Then we construct $\{\mathbf{l}_m\}$ passing through $\{u_m\}$. The direction of $\{\mathbf{l}_m\}$ is orthogonal to $\mathbf{c}(u)$'s tangent vector and the input shape's face normal. We set the length of $\{\mathbf{l}_m\}$ to d_{bb} and the center of $\{\mathbf{l}_m\}$ at $\{u_m\}$.

Then we generate a ruled surface $\mathbf{s}$ to fit $\{\mathbf{l}_m\}$. We use the same representation of ruled surfaces in Formula (5). So we denote $\mathbf{c}_0(u)$ and $\mathbf{c}_1(u)$ as the two B-splines on two sides of $\mathbf{s}$. Their configuration of control points and knot vectors are the same as $\mathbf{c}(u)$'s. We denote the start and end points of $\mathbf{l}_m$ as l_m^s l_m^e respectively. The fitting problem can be formulated as follows:

$$\arg\min_{C_0,C_1} \left(\sum_{m=0}^{M} \|\mathbf{c}_0(u_m) - l_m^s + \mathbf{c}_1(u_m) - l_m^e\|_2^2 + \beta \sum_{0\le i\le j\le n} \|\mathbf{r}_i - \mathbf{r}_j\|_2^2 \right), \tag{B.2}$$

where C_0 and C_1 are the control points of $\mathbf{c}_0(u)$ and $\mathbf{c}_1(u)$, $\mathbf{r}_i$ is the difference between the corresponding control points in C_0 and C_1. We set $\beta = 50$ in our experiments. This problem is easy to solve by using least squares.

Step 3: We perform the PCA method to generate a plane from the ruled surfaces $\mathbf{s}$ we get in **Step** 2 and then move it along its normal to find a collision-free place. To ensure the plane is long enough to cover the targe patch, we set its length 1.2 times $\mathbf{c}(u)$'s. Finally, we uniformly sample along the two perpendicular edges to the ruling direction to generate several control points on each side, with the number of control points on each side being the same as $\mathbf{c}(u)$'s, thus providing a feasible initialization.

Appendix C. Parameter list

Table C.3: Summary of parameters used in our method

Parameter	Symbol	Value
difference of two principal curvatures	χ	10
diagonal length of the bounding box	d_{bb}	-
average edge length	d_{ae}	-
length of a ruling line	β	d_{bb}
a small distance	δ	$0.25d_{\text{ae}}$
number of sampling triangles	N_{seed}	200
number of sampling points on each direction line $\mathbf{L}$	N_{sample}	200
distance of translation	$d_{\text{translation}}$	$1.5\%d_{\text{bb}}$
minimum projection distance	d_{cover}	$2.5\%d_{\text{bb}}$
the length of the ruled surface strip	γ	$5d_{\text{ae}}$
number of subdivisions in the guide curve direction	k	8
number of subdivisions in the ruling direction	l	7
maximum iteration number	N_{iter}	10
number of rotation	N_{rotate}	15
approximation error bound	ϵ	$2\%d_{\text{bb}}$
smoothness parameter	ω_{uv}	5
segmentation parameter	ω_{seg}	0.5