Adaptive Slicing Based on Efficient Profile Analysis

Huachao Mao\textsuperscript{a}, Tsz-Ho Kwok\textsuperscript{a}, Yong Chen\textsuperscript{b,}\textsuperscript{*}, Charlie C.L. Wang\textsuperscript{b}

\textsuperscript{a}Epstein Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089, USA
\textsuperscript{b}Department of Design Engineering, Delft University of Technology, The Netherlands

Abstract
Adaptive slicing is an important computational task required in the layer-based manufacturing process. Its purpose is to find an optimal trade-off between the fabrication time (number of layers) and the surface quality (geometric deviation error). Most of the traditional adaptive slicing algorithms are computationally expensive or only based on local evaluation of errors. To tackle these problems, we introduce a method to efficiently generate the slicing plans by a new metric profile that can characterize the distribution of deviation errors along the building direction. By generalizing the conventional error metrics, the proposed metric profile is a density function of deviation errors, which measures the global deviation errors rather than the in-plane local geometry errors used in most prior methods. Slicing can be efficiently evaluated based on metric profiles in contrast to the expensive computation on models in boundary-representation. An efficient algorithm based on dynamic programming is proposed to find the best slicing plan. Our adaptive slicing method can also be applied to models with weighted features and can serve as the inner loop to search the best building direction. The performance of our approach is demonstrated by experimental tests on different examples.

Keywords: Additive Manufacturing, Adaptive Slicing, Geometric Profile, Sampling

1. Introduction
Over the last thirty years, a new type of manufacturing process, called additive manufacturing (AM), has been developed using the principle of layer-based material accumulation [1]. Many novel AM processes based on different techniques such as laser curing, nozzle extrusion, jetting, electron beam, and laser cutter, have been developed [2]. AM is a direct manufacturing process that can fabricate parts directly from computer-aided design (CAD) models without part-specific tools or fixtures. Therefore, it can fabricate highly complex parts effectively. In most of the AM processes, the digital CAD model is sliced by intersecting it with a number of horizontal planes. The sliced contours are then transferred to generate the tool paths for material accumulation. While the uniform layer thickness is widely used due to its simplicity, it has been theoretically proven that adaptive layer thickness can produce parts with higher accuracy and shorter building time [3, 4]. In adaptive slicing, the varied thickness of layers is determined by the geometry of input models. Most of the existing methods [4] evaluate deviation errors locally by the geometry at particular slicing planes, which can result in large approximation error when there is complex geometry between neighboring slices.

To improve the accuracy of geometry error evaluation, different strategies have been developed, including: 1) slicing the model using the finest layer thickness [5], 2) direct slicing the designed CAD model (like NURBS [6], CSG [7]) rather than the related tessellated model (like STL [3]), and 3) refining the slicing plan if sharp geometry changes are detected within one layer [8]. However, these methods are computationally expensive, and the bottleneck is caused by the process requiring a large number of intersection operations between the slicing plane and the CAD model.

In this paper, we develop a novel method to overcome this bottleneck by representing a CAD model as a profile of the geometric error along the building direction. An example is shown in Fig. 1. The evaluation of the geometric deviation error can be done on the profile rather than intersecting the CAD model intensively. By constructing the metric profiles using GPU-accelerated methods, the whole process of adaptive slicing can be computed in about one second. In summary, our presented adaptive slicing technique is accurate and efficient. The main contribution is a new profile-based framework for adaptive slicing, which shows the following properties.

1. Global: The profile generalizes the conventional error metrics and provides an implicit representation for the shape of an input model, and it is global information for slicing. Based on that, we design an optimization algorithm based on dynamic programming to find the best slicing plan.
2. Efficient: The analysis taken in our algorithm is based on a metric profile that can be generated by GPU-accelerated techniques. In our tests, the whole process from profile construction to getting the op-
timal slicing plan can be completed in around one second. Benefiting from the efficiency, it can serve as the inner loop to find the best building direction.

3. General: The formulation can be easily extended to integrate different commonly used error metrics. Moreover, we also show that it can be further generalized to incorporate the user specified salience.

The rest of the paper is organized as follows. Section 2 briefly describes the related work of adaptive slicing in additive manufacturing. After that, the details of our framework are given in Section 3. Section 4 demonstrates the generality of the presented framework by considering different factors. Results and statistics are given in Section 5, and our paper ends with conclusions and discussions in Section 6.

2. Related Work

Layer-based additive manufacturing fabricates a part by successively accumulating material layer by layer, in which an essential computational step is slicing. In the step of slicing, the input CAD model is intersected with a set of horizontal planes, and this results in a set of closed curves or polygons at different height levels. Assuming each layer is fabricated by extruding the intersected contour with a small layer thickness, such an extrusion introduces the staircase error [9], which is directly related to the surface angle and the layer thickness [10]. Slicing methods can be classified into uniform (having an equal thickness in all layers) or adaptive ones (with unequal layer thicknesses in different layers). While uniform slicing is simple and fast, adaptive slicing is proven to be able to fabricate parts with higher accuracy and shorter building time.

2.1. Surface quality

Various slicing procedures have been discussed in previous surveys of AM technology [11, 12], where the resultant quality is measured by different geometric errors – e.g., cusp height, surface roughness (Ra), and area or volumetric deviation. The most widely used error measurement is the cusp height [3]. Kulkarni and Dutta [4] reduced the staircase effect by controlling the maximum allowable cusp height using 12 different expressions. The relationship between the maximal allowed cusp height and the normal vector at any point on the tessellated model is used to find the thickness of each layer. Yan et al. [13, 14] followed this idea and developed an adaptive slicing method that works directly on point cloud by using moving least square surfaces. This error measurement has also been used widely in different applications [15, 16, 17, 18, 19]. Recently, Wang et al. [20] presented an adaptive slicing method considering both the cusp height and saliency criterion. The Ra value [8], which is commonly used in design or manufacturing practice to specify surface roughness, is a similar measurement that can be used as well. Differently, Zhao [7] introduced area deviation by comparing the measured deviation of the interior area of the layer contours to check whether the layer becomes thicker or thinner. However, a staircase effect appears on the layer while two contours have a similar area but totally different shapes. All of these errors can be classified as 2D measurements. They become less suitable when the geometry of an input model between two neighboring slices becomes more complex. To address the problem, Kumar and Choudhury [21] presented a volume deviation for adaptive slicing. This technique is a promising solution for slicing CAD models with remarkable higher precision. However, since it works directly with surfaces of the part to mathematically compute the related volumes, the geometric complexity of the surfaces would need some complicated mathematical computation that may jeopardize the validity of such system.

2.2. Global slicing

Most of the previous work (e.g., [22]) first cuts the entire part from the bottom-most to the top-most position at the maximal thickness that is allowed by the AM process, and then applied the specific error to decide if some thicker lay-
ers need to be further sliced into thinner ones. However, the presence of any concave or convex area may yield a significant geometry deviation error when no staircase effect is identified at either slicing layers. In contrast, Hayasi and Asiabanpour [5] started slicing at the minimal allowed thickness, then allowed the current layer becomes thicker or thinner while comparing the obtained error with the given tolerance. Singhal et al. [8] presented a comprehensive and more accurate direct slicing procedure by detecting the sharp concave/convex vertices and then subdividing a large layer into a number of thinner layers if sharp concave/convex vertices are detected. Wang et al. [20] proposed an iterative method to refine the slicing plan obtained by previous greedy methods. All these slicing procedures are trying to obtain the geometry’s sharp changes as accurately as possible, either by cutting at minimum thickness or by checking the sharp vertices. However, the expensive computation prevents to applying this methodology at very high resolution. To solve this dilemma, we introduce a slicing algorithm that optimizes the slicing plan on a profile, which can be efficiently constructed from the CAD model.

### 3. Adaptive slicing based on metric profiles

The complexity of the slicing optimization problem mainly arises from the evaluation of geometry error. This geometry error calculation is time consuming because most slicing algorithms are based on NURBS [6], STL [3] or point cloud set [14], and slicing one 3D model based on these representations is computationally expensive. Instead of directly evaluating the deviation error on input CAD models, we propose an intermediate metric profile to evaluate the deviation error distribution along the z axis (the printing direction). The value of metric profile $\phi(z)$ is a measure of geometric error density with reference to the height $z$. Based on the definition of metric profile $\phi(z)$, we can evaluate both the error metric of each layer and the total error with trivial effort. Specifically, the metric error $\varepsilon_k$ of a layer $k$ is defined as the integral of a metric profile function along that layer’s height range $[z_{k-1}, z_k]$ as

$$\varepsilon_k = \int_{z_{k-1}}^{z_k} \phi(z)dz,$$  

(1)

**Problem Definition:** Based on the metric profile (error density function) $\phi(z)$ and given the allowed maximal error $\epsilon$ of a layer, the optimization objective is to minimize the total number of layers while assuring each layer’s integral error $\varepsilon_k$ is within the given tolerance $\epsilon$, i.e.

$$\min_K \quad \text{s.t.} \quad \varepsilon_k = \int_{z_{k-1}}^{z_k} \phi(z)dz \leq \epsilon, \quad k = 1, ..., K \quad (C1)$$  

$$z_k = z_{k-1} + t_k, \quad k = 1, ..., K \quad (C2)$$  

$$t_{\text{min}} \leq t_k \leq t_{\text{max}}, \quad k = 1, ..., K \quad (C3)$$  

$$z_0 = 0, \quad z_K = H \quad (C4)$$

where $\phi(z)$ is the metric profile function, the unknown variable $K$ is the number of layers, the unknown variable $t_k$ is the thickness of layer $k$, $t_{\text{min}}$ and $t_{\text{max}}$ are the manufacturing constraints of the minimal and maximal layer thickness, and $H$ is the height of the input model.

To solve this slicing problem, we propose a novel adaptive slicing pipeline as shown in Fig. 2. Different from the traditional adaptive slicing pipeline that directly performs the slicing on the CAD model (refer to the bottom row of Fig. 2), our new pipeline first efficiently samples the input 3D model into structured points and then constructs the metric profile from the sampled points. Using the metric profile, we can efficiently obtain the optimal slicing plan, and eventually export the tool paths for 3D printers such
as Fused Deposition Modeling (FDM) [23] or projection-based Stereolithography (SLA) [24]). The details of each step in our new framework will be discussed as follows.

3.1. Metric Profile

We introduce “metric profile” $\phi(z)$ to describe the geometry error distribution along the z direction (the printing direction). The metric profile is a function to measure the geometry error density at height $z$. There are different error metrics can be used to construct the profile as discussed in Section 2.1, e.g., cusp height, surface roughness, area deviation and volume deviation. The proposed framework is compatible and useful for all these errors and other applications as long as a metric profile can be defined. In this section, as a widely used metric since its development [3], the cusp height is picked to explain the algorithm. In Section 4.1, we will demonstrate that a similar setting can be developed for other error metrics, as well as the weighted saliency $\omega(z)$ in Section 4.2.

The error metric “cusp height” is to measure geometry error due to the lack or surplus of materials caused by slicing. The cusp height $c$ of the $i^{th}$ layer is calculated by its thickness $t_i$ and the normal of the points in the layer. Assume the slicing is along the z direction and the maximum value of normal in z-axis is $n_z$ (from a normal vector $n = (n_x, n_y, n_z)$), the cusp height of the layer can be approximated by

$$c = \|t_i \cdot n_z\|_{\infty}.$$  \hfill (3)

As there are many points with different normal values in a layer, the infinity norm is used (i.e., the maximum value of $n_z$ is picked as the error density) to conservatively preserve the sharpest feature. Remarked that, the metric of cusp height is used here just for the sake of explaining the algorithm, but same concepts can be applied to other metrics.

By definition, the thickness $t_i$ is always along the printing direction $z$, so a change in the value of $t_i$ can be denoted as $\Delta z$. Thus the corresponding change in cusp height is $\Delta c = \Delta z \|n_z\|_{\infty}$. Therefore, the metric profile for cusp height is defined as the derivative of $c$ w.r.t. the height $z$:

$$\phi_{\text{cusp}}(z) = \frac{dc}{dz} \approx \frac{\Delta c}{\Delta z} = \max\{n_z\}.$$  \hfill (4)

Based on the above-defined error metric profile, we can easily calculate the metric error $\varepsilon_k$ of a layer $k$ within the height range $[z_{k-1}, z_k]$ as the integral of metric profile $\varepsilon_k = \int_{z_{k-1}}^{z_k} \phi(z)dz$. Noted that, this proposed integral error metric is a generalization of the conventional slicing algorithm using cusp height. The illustration and comparison of the approximation errors between the conventional and the proposed methods are given in Appendix A. In Section 4.1, we will see the concept of “metric profile” could be extended to other commonly used geometry error metrics, such as surface roughness, area deviation and volume deviation.

3.2. Sampling for Profile Construction

To improve the efficiency of the implementation, the entire range of the metric profile $\phi(z)$ is divided into a series of intervals. The bins (intervals) are consecutive and non-overlapping intervals of the height $z$, and they have equal size of $b$, i.e., $\phi(z) = \{\phi(1), ..., \phi(N)\}$ where $N = \lfloor \text{model height} / \text{interval size} (b) \rfloor$ is the number of bins, and the size of each interval $b$ is set as a small value (2μm in our test cases; in comparison, the layer thickness typically used in SLA is 100μm, and the choice of $b$ will be discussed in Sec. 5). We employ the sampling techniques to facilitate the computation of the metric profile value in each interval. It can be well-structured points, voxels, or rays, and the geometric error could be easily evaluated by checking all the sampled points falling into the corresponding interval. This process is efficient compared to the expensive intersection operation based on the original CAD model.

In this paper we choose the Layer Depth Image (LDI) as our sampling approach. LDIs [25, 26, 27] is an extension of the ray representation (ray-rep) in solid modeling. Based on a well-structured discrete sampling approach, LDIs can efficiently and robustly perform a set of complex geometric operations, including offsetting [28, 29], Boolean [30], regulation [31] and overhang area evaluation [32]. By parallel GPU computing, LDI could achieve high resolution efficiently (a STL model with 1 million faces can be sampled into LDI within one second). Comparing to directly slicing a model with similar complexity in the finest resolution that takes several minutes, it saves a lot of computation time. Generally, the resolution of the LDI is dense enough for normal models. For a bigger size model or a higher accuracy is needed, a technique called volume tiling [31] can be used. That is, the bounding box of a model is first split into smaller tiles. Each tile is then processed independently and we construct their LDI models respectively. Besides the sampling efficiency, LDI is also a rich sampling representation (denoting a sampled model as $P = \{p_j\} = \{x_j, y_j, z_j, n_x^j, n_y^j, n_z^j, f_j, r_j\}$), which includes the point coordinate $(x_j, y_j, z_j)$, normal $(n_x^j, n_y^j, n_z^j)$, ID of facet $f_j$ where the point belongs to, and ID of sampling ray $r_j$ that has the information of point adjacency and In/Out specification (i.e., the intersection point where the ray goes into or gets out from the model).

Among the information, the point coordinate $(x_j, y_j, z_j)$ is always useful, and the facet ID $f_j$ is used to retrieve the weight $\omega_j$. In addition, the normal $(n_x^j, n_y^j, n_z^j)$ is used to construct the profiles of cusp height and surface roughness, while the ID of sampling ray $r_j$ is used to build the profiles of area deviation and volume deviation. The ray $r_j$ is also useful in the generation of toolpaths/mask-images for fabrication.

For example, the cusp metric profile in eq.(4) requires the normal $(n_x^j, n_y^j, n_z^j)$. The construction of the metric profile can be performed efficiently by grouping the points into the corresponding intervals according to their heights.
Recall that the metric profile of cusp height in Eq.(4) is discretized into N intervals, and for each interval $i$, the metric value $\tilde{\phi}_{cusp}(i)$ is the maximal value of $n_z$ among all the points inside the interval, i.e.,

$$\tilde{\phi}_{cusp}(i) = \max \{ n_z^i \} \big| z/b = i, P_j \in P \} ,$$  \hspace{1cm} (5)

Hence, the metric profile value at height $z$ could be approximated as

$$\phi_{cusp}(z) \approx \tilde{\phi}_{cusp}(\lfloor z/b \rfloor) .$$  \hspace{1cm} (6)

This profile construction process can be done within 50 ms on GPU for all our tests.

Section 4.1 will introduce some other metric profiles, which can also be constructed using the LDI information, and the detailed usage will be discussed there.

### 3.3. Slicing Algorithm

After constructing the metric profile $\phi(z)$, we have all the information to formulate the slicing optimization problem of Eq.(2). We represent the slicing plan (S) as a boolean array with a size equal to the number of intervals $N + 1$, where the value of true or false stands for if there is a slice on that particular height or not. There are manufacturing constraints of the minimal and maximal layer thickness $[t_{\text{min}}, t_{\text{max}}]$, and the size of interval $b$. As the intervals are used to optimize the location of the slicing planes, if $b$ is a common factor of $t_{\text{min}}$ and $t_{\text{max}}$, it is possible for the algorithm to utilize the whole range of layer thickness. Otherwise, the range of the layer thickness will be narrowed down, and the minimal and maximal number of intervals in one layer will be $\lfloor t_{\text{min}}/b \rfloor$ and $\lfloor t_{\text{max}}/b \rfloor$.

To find a slicing plan, one may apply a greedy algorithm to assign as many intervals as possible to a layer until the sum of metric profile $\phi(i)$ exceeds the tolerance $\epsilon$. However, a greedy heuristic may yield locally optimal solutions and fail to satisfy the constraints while the minimal layer thickness is restricted. Although it is faster to calculate, we find that this kind of situation is not rare and appears from time to time. Moreover, such a greedy algorithm cannot guarantee the result with a minimal number of layers. It motivated us to design an efficient algorithm based on Dynamic Programming (DP) [33] to compute a true global optimum. For a bottom-to-top DP algorithm, it starts from the head and computes sub-solutions from smaller to bigger problems, and then stores the intermediate results in the memory. These previously computed solutions are combined to give the best solution for the whole problem. Once it has reached the tail, the optimal slicing plan can be extracted by backtracking.

The process is illustrated using an example of first 8 intervals $\{\tilde{\phi}(1), ..., \tilde{\phi}(8)\}$ shown in Fig. 3. Assume the allowable error of a layer is $\epsilon = 0.6$, and the minimal and maximal number of intervals in a layer are 2 and 3 to satisfy the constraint C3 in Eq.(2), i.e., $\lfloor t_{\text{min}}/b \rfloor = 2$ and $\lfloor t_{\text{max}}/b \rfloor = 3$. A weight array $K[0 \ldots 8]$ and an index array $D[0 \ldots 8]$ for the slicing planes, i.e., $K[1]$ is in between $\tilde{\phi}(1)$ and $\tilde{\phi}(2)$. The weight array $K[\ldots]$ stores the least number of layers and the index array $D[\ldots]$ records the index of optimal slicing positions, which will be used in backtracking.

Starting from the head ($0^{th}$), it is initialized with $K[0] = 0$ and $D[0] = NA$ as the first slice plane to satisfy C4 in Eq.(2). The $1^{st}$-plane is initialized with $K[1] = \infty$ and $D[1] = NA$ as for the minimal number of intervals to form a layer is 2, it is not allowed to put a slice plane here. For the $2^{nd}$-plane, it can form a layer only with $0^{th}$-plane including $\tilde{\phi}(1)$ and $\tilde{\phi}(2)$, so $K[2] = K[0] + 1 = 1$, where the ‘+ 1’ means one layer, and $D[2] = 0$, meaning $0^{th}$-plane. In other words, if there is a slice plane on $2^{nd}$-plane, this layer is optimal forming by the $0^{th}$-plane and itself. As the minimal and maximal number of intervals are 2 and 3, there are two possible positions to form a layer with the $3^{rd}$-plane, which are the $0^{th}$- and $1^{st}$-plane. A comparison can be made to find the optimal one, and this sub-solution will be stored in the memory. Specifically, two cases of forming a layer with the $0^{th}$-plane including $\tilde{\phi}(1)$ to $\tilde{\phi}(3)$ and with $1^{st}$-plane including $\tilde{\phi}(2)$ to $\tilde{\phi}(3)$ are compared, and the minimum one is picked. For $0^{th}$-plane, it is $K[0] + 1 = 1$; and for $1^{st}$-plane, it is $K[1] + 1 = \infty + 1 = \infty$. In this case the $0^{th}$-plane wins, and this sub-solution is stored as $K[3] = 1$ and $D[3] = 0$. Similarly, the $4^{th}$-plane will store $K[4] = 2$ and $D[4] = 2$. For the $5^{th}$-plane, the two cases are with the $3^{rd}$-plane and the $4^{th}$-plane. However, both $\tilde{\phi}(3) + \tilde{\phi}(4) + \tilde{\phi}(5) = 0.9$ and $\tilde{\phi}(4) + \tilde{\phi}(5) = 0.7$ exceeds the allowable error of a layer $\epsilon = 0.6$. Therefore, none of them is feasible, so the weight is set as $K[5] = \infty$ and $D[5] = NA$ to prevent a plane is placed at this position. The iteration continues, and the rest are $K[6] = 3$ and $D[6] = 4$; $K[7] = \infty$ and $D[7] = NA$; $K[8] = 4$ and $D[8] = 6$.

The key of the DP algorithm is making use of the stored sub-solutions (e.g., $K[0 \ldots 7]$) to compute the later ones until the last ($K[8]$). Once it has been done, the optimal slice plan can be extracted by backtracking using the index array $D[0 \ldots 8]$. Starting from $D[8]$, which records the optimal one to form a layer should be the $6^{th}$-plane, and recursively $D[6]$ returns 4, $D[4]$ returns 2, and $D[2]$ returns 0, which reaches the head. As a result, the slice plan $S[0] = S[2] = S[4] = S[6] = S[8] = True$, and there are four layers: $\phi(1) \phi(2)$, $\phi(3) \phi(4)$, $\phi(5) \phi(6)$, and $\phi(7) \phi(8)$. The slice plan is valid and each of the layers

<table>
<thead>
<tr>
<th>Metric Profile</th>
<th>$\phi(1)$</th>
<th>$\phi(2)$</th>
<th>$\phi(3)$</th>
<th>$\phi(4)$</th>
<th>$\phi(5)$</th>
<th>$\phi(6)$</th>
<th>$\phi(7)$</th>
<th>$\phi(8)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SP^#$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$K[i]$</td>
<td>0</td>
<td>$\infty$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>$\infty$</td>
<td>3</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$D[i]$</td>
<td>NA</td>
<td>NA</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>NA</td>
<td>4</td>
<td>NA</td>
</tr>
</tbody>
</table>

$S[i]$ - slice plane number, $S[i]$ - the resulted optimal slicing plan.

Figure 3: An example with the first 8 intervals in the metric profile is used to illustrate the Dynamic Programming based slicing process. $SP^\#$ - slice plane number.
has an error smaller than the allowable error, so the DP algorithm can find a global optimum where the greedy algorithm probably will take the first three intervals \((\hat{\phi}(1) \hat{\phi}(2) \hat{\phi}(3))\) and get a local optimum for the first layer, and then being stuck with \((\hat{\phi}(4) \hat{\phi}(5))\) as the sum of them (0.7) is greater than the allowable error \(\epsilon = 0.6\).

Note that, if there is no feasible solution, the tail will store invalid values, e.g., \(K[8] = \infty\) and \(D[8] = NA\), and the system will report and ask for a revision of the constraints. Otherwise, the backtracking will never reach the head.

A general version of the slicing algorithm is summarized in Procedure 1. It takes a linear time \(O(N)\) to compute and track the optimal slicing plan, where \(N\) is the number of intervals (the worst case is quadratic time \(O(N^2)\) for the range of minimal and maximal interval number in a layer equals to the total number of intervals). In all of our test cases, this slicing algorithm takes less than 5 ms.

Procedure 1 Slicing Based on Dynamic Programming

Input: Metric profile intervals \(\{\hat{\phi}(1), \ldots, \hat{\phi}(N)\}\), interval size \(b\), allowable error for a layer \(\epsilon\), max/min layer thickness \((t_{\text{min}}, t_{\text{max}})\)

Output: Slicing Plan \(S[0 \ldots N]\)

//initialization
1: \(S[0 \ldots N] = false; K[0 \ldots N] = \infty, D[0 \ldots N] = NA\)
2: \(K[0] = 0; //\text{for the 0}^{th}\text{-plane}\)

//bottom-to-top dynamic programming
3: for \(i = 1\) to \(N\) do
4: for \(l = i - \lfloor t_{\text{max}}/b \rfloor\) to \(i - \lfloor t_{\text{min}}/b \rfloor\) & \(l \geq 0\) do
5: if \(\sum_{m=1}^{i} \hat{\phi}(m) \leq \epsilon\) then
6: if \(K[i] \geq K[l] + 1\) then
7: \(K[i] = K[l] + 1; D[i] = l;\)
//backtracking
8: \(i = N;\)
9: while \(i > 0\) do
10: \(S(i) = true;\)
11: \(i = D(i);\)

The optimization based on the integral of metric profile in Eq.(1) works very well in the general models as will be shown in the result section. Moreover, we can optionally insert some must-slicing plane heuristically. For some special cases that a general flat metric profile has some sharp changes, the integration may not exceed the allowable limit even a sharp change is included, because its neighborhood is very low in value. For instance, the CAD model shown in Fig. 4 has two such kind of sharp changes in the profile (marked as A and B) due to the transitions between the cube and the cylinder. Fortunately, the sharp changes can be easily detected in the profile, and thus the special case can be easily handled. We introduce one more step before applying the optimization, and it is simply to go through the profile from the bottom to the top and compute the difference for each consecutive interval. If the difference is greater than a threshold (i.e., 0.5), a slice is placed in the upper interval, and the profile is separated into two segments by the slice. After that, the optimization can be performed separately in different segments, and they are assembled to a final result. Figure 4 has shown the detail of this step.

Figure 4: Based on the metric profile of a given CAD model, sharp changes (located at A and B, which are the sharp transitions between the cube and the cylinder) can be detected. Our algorithm can successfully place slicing planes on these positions.

3.4. Tool Path/Image Generation

After obtaining the optimal slicing plan, we can generate the projection images for SLA printers, or the tool paths for FDM printers. Traditionally, the contours for the images are directly computed from the intersection between the \(z\) planes and the CAD model [34], which usually needs triangulation of the contour to generate the images. On the contrary, because LDI is a kind of ray representation, we can easily determine whether a certain pixel is black/white by checking the position of this pixel [35]. Specifically, given that the resolution of the image is the same as that of LDI in the \(z\) direction, for any height of an image plane, we can go through every ray of the LDI in each pixel and find the least sampling point that is greater than the given height. After that, the black/white of the pixel can be determined by the In/Out specification of the sampling point. As the sampling point is the next intersection point along the ray in the \(z\) direction, if it is specified as Out, then the position of the pixel has to be inside the model and the pixel should be white, and vice versa. This can generate the images in a very efficient way. The tool paths are then designed by the sequence for accumulating material to fill up the whole image.

4. Other Factors

Our slicing pipeline is a general slicing framework based on sampling approach. In this section, we extend our adaptive slicing framework by considering other factors rather than cusp heights: 1) other commonly used error metrics are shown to be compatible with our framework, 2) the mesh saliency map is integrated into the metric profile to preserve the salient features, and 3) the proposed slicing algorithm can be served as a building block in searching the optimal fabrication direction efficiently.
4.1. Different Metric Profiles

In physical modeling, the commonly used geometry deviation error metrics includes cusp height, surface roughness, area deviation, volume deviation, etc. We show that our framework is general to incorporate these metrics. Figure 5 shows all the metric profiles and their corresponding slicing plans. We have already demonstrated our framework using cusp heights, and we will briefly describe the details of formulation for others in below.

4.1.1. Surface Roughness

Singhal et al. [8] figured out a statistic model for predicting the surface roughness (Ra), and the overall metric is the averaging surface roughness of the whole part,

\[ Ra = \frac{t}{\sin \theta}, \]

where \( \theta \) is the building angle between the building direction and surface normal, and \( t \) is the layer thickness.

Similar to the derivation of metric profile using cusp height in Sec. 3.1, surface roughness metric profile can be expressed as the derivative of surface roughness error:

\[ \phi_{Ra}(z) = \frac{dRa}{dz} = \lim_{\Delta z \to 0} \frac{\Delta Ra}{\Delta z} = \frac{1}{\sin \theta}, \]

where \( \phi_{Ra}(z) \) returns the maximum value of \( \sin^{-1} \theta \) for all the points at height \( z \).

In the software implementation, given a sampled model \( P = \{p_j\} \), we know the point coordinate \((x_j, y_j, z_j)\), and the normal \((n^x_j, n^y_j, n^z_j)\) of each point \( j \), and the angle \( \theta \) between point normal and fabrication direction satisfies \( \cos \theta = n^z_j \). Thus, the maximal surface roughness of the interval \( i \) can be computed as

\[ \tilde{\phi}_{Ra}(i) = \max \left\{ \frac{1}{\sqrt{1 - n^2_j}} \|z_j/b\| = i, p_j \in P \right\}. \]

Hence, the metric profile value for surface roughness at height \( z \) could be approximated as

\[ \phi_{Ra}(z) \approx \tilde{\phi}_{Ra}([z/b]). \]

4.1.2. Area Deviation

Zhao [7] presented area deviation as a new surface quality metric for their adaptive slicing algorithms. The area deviation is defined as the relative deviation of top area and bottom area of a layer,

\[ \delta_A = \frac{|A_{z} - A_{z+k}|}{A_{z}}, \]

where \( A_z \) is the area of slicing at \( z \) plane, and \( A'_z \) is the derivative of \( A_z \).

The computation of the area in a particular height \( z \) is similar to the image generation described in Sec. 3.4. The area of a layer is computed by the number of pixels that are specified as white in the image plane at the specific height, i.e., assuming there are \( N_w \) pixels are in white, the area is approximated as

\[ A_z \approx N_w \cdot w^2, \]

where \( w \) is the size of the pixel.

We have also generated a series of intervals having exactly the same size of the metric profile’s one for storing the areas in different heights \( A_z = \{\tilde{A}(1), \ldots, \tilde{A}(N)\} \). The height value used to compute the area for each interval is taken as the center of the interval. Thus the area deviation of the interval \( i \) can be computed as

\[ \phi_{area}(i) = \frac{|\tilde{A}|}{A(i)} = \frac{|\tilde{A}(i + 1) - \tilde{A}(i)|}{bA(i)}. \]

4.1.3. Volume Deviation

Instead of checking only one-dimensional slicing error (cusp height) or two-dimensional slicing error (area deviation), Kumar and Choudhury [21] proposed a cusp volume metric to quantify the three-dimensional error, which is the volume error between the CAD model and the printed part. The volume of the \( k \)-th layer with the height range \((z_{k-1}, z_k)\) in the CAD model can be calculated as

\( V_{CAD} = \int_{z_{k-1}}^{z_k} A_z \cdot dz \). The corresponding layer volume in the printed part is an extrusion with its bottom area \( A_{z_{k-1}} \), i.e., \( V_{printed} = A_{z_{k-1}} \cdot t_k \). Hence, the volume deviation of the \( k \)-th layer is

\[ \delta_V = \frac{V_{CAD} - V_{printed}}{V_{printed}} = \frac{\int_{z_{k-1}}^{z_k} A_z \cdot dz - A_{z_{k-1}} \cdot t_k}{A_{z_{k-1}} \cdot t_k}. \]

Rewriting it as a differential form for a particular height \( z \) in the \( k \)-th layer gives that

\[ \frac{d\delta_V(z)}{dz} = \frac{d}{dz} \int_{z_{k-1}}^{z_k} (A_z - A_{z_{k-1}})dz = A_z - A_{z_{k-1}}. \]

However, as the volume deviation is a height-dependent error metric, we cannot directly define the above equation as the metric profile. This is because the term \( A_{z_{k-1}} \) is not an independent term, but a specific term for a particular layer. Fortunately, our optimization framework based on dynamic program can handle this nonlinear problem with only a little modification. By defining the volume deviation profile as

\[ \phi_{vol}(z) = A_z, \]

the metric error of the \( k \)-th layer can be computed as

\[ \epsilon_k = \int_{z_{k-1}}^{z_k} (\phi(z) - \phi(z_{k-1}))dz. \]
Replacing the error computation in Eq.(2) by this one, the slicing algorithm presented in Procedure (1) works exactly the same for the area deviation. The computation of the area $A_z$ is the same as Eq. (13), and thus the discrete form for Eq.(17) is

$$
\tilde{\phi}_{vol}(i) = \tilde{A}(i),
$$

with

$$
\tilde{e}_k = \sum_{j=l+1}^{i} |\tilde{\phi}(j) - \tilde{\phi}(l+1)| \cdot b.
$$

### 4.1.4. Comparing different error metric profiles

Figure 5 compares metric profiles and slicing results of different error metrics. The metric profiles reflect the features of the users’ interest. For example, the volume deviation metric cares about the portion that has a large amount of volume error, and the bottom half part has a large volume and thus the bottom half of error metric profile is larger than up half part’s. In comparison, the cusp height metric measures the cusps which is not related to the contour size, and hence it has relatively balanced metric profile compared to the volume deviation metric profile.

The exemplary “hearing aid” model in Fig. 5 has surfaces facing up around the middle height. These surfaces introduce large staircase error. We also notice that all these error metric profiles have the larger error density value at the height of these surfaces. Accordingly, all the four optimal slicing results have a smaller layer thickness at the height of these surfaces, i.e. the red lines in the middle height.

Similar to the cusp height in Eq.(4), the error metrics surface roughness in Eq.(8) and area deviation in Eq.(12) are formulated as the integral of density function $\phi(z)$, which only depends on the particular height $z$. Therefore, all the three error metrics have the additive property: the error for a thick layer is the sum of the errors evaluated on the thinner layers that are combined into the thicker layer. However, the formulation Eq.(18) indicates that this additive property does not hold for the volume deviation, because the error density function depends not only the information of height ($z$) but also the base height of that layer ($z_{k-1}$). Fortunately, the “volume deviation” for any layer can be easily computed using the Eq.(18), with underlying precomputed profile $\phi_{vol}(z) = A_z$.

### 4.2. Slicing of Weighted CAD model

In some cases, the features of a CAD model are not equally important, and a straightforward idea is that the more salient features will be sliced with a smaller layer thickness. For example, in the CAD model of Fig. 6, the face of David model is considered as more salient than the rest part. By adding a weight (whose value ranges from 0 to 1) to each face, we can adjust the importance of each feature (face). In this paper, two methods of generating the weights are combined. Lee [36] introduced a multi-scale mesh saliency computation method, which is the basis of our salient mesh. We also develop an interactive interface for users to explicitly assign saliency level to each mesh facet or region. Then, a combined mesh weight map will be generated by the per-face product of geometry and user-specified saliency. An example can be found in Fig. 6. The saliency map for geometry-based method has high weight for small features along the $z$ axis, while by specifying the high weight only in the face of David model, we can further focus our saliency weight onto the portion that the user prefers.
Recall that the LDI sampled points has the following information \( P = \{ p_j \} = \{ x_j, y_j, z_j, n_j^x, n_j^y, n_j^z, f_j, r_j \} \). After each face has been assigned a weight value, each sampled point \( p_j \) can inherit the weight value from its belonged face. Denote each point’s weight as \( \omega(p_j) \). When computing the metric profile, each point’s weight can be applied point-wisely to modify the metric profile. For example, the metric profile with “cusp height” considering weights can be demonstrated in the result section, our proposed slicing algorithm is very efficient that can complete the slicing on different criterion, e.g., visual saliency [20], perceptual model [32], mechanical strength [37], etc. As will be demonstrated in the result section, our proposed slicing algorithm is very efficient that can complete the slicing plan for all the models within one second. Such efficiency plays an important role in finding the optimal fabrication orientation, which requires a large number of slicing evaluations along different orientations (normally about 1000 direction candidates).

4.3. Building Direction

Determining an optimal printing orientation can further improve the printing performance, and it can be based on different criterion, e.g., visual saliency [20], perceptual model [32], mechanical strength [37], etc. As will be demonstrated in the result section, our proposed slicing algorithm is very efficient that can complete the slicing plan for all the models within one second. Such efficiency plays an important role in finding the optimal fabrication orientation, which requires a large number of slicing evaluations along different orientations (normally about 1000 direction candidates).

In this paper, we present a nested loop to find the optimal building orientation based on the fabrication time and surface quality. In the outer loop, we uniformly sample the orientation space into 1000 directions (Fig. 8(a)). For each orientation, given the allowable maximum cusp...
height, we run our adaptive slicing algorithm as the inner loop, and obtain the optimal slicing plan with the corresponding fabrication time and surface quality. We use the weighted summation of fabrication time and surface roughness as the objective energy function:

\[ F(O) = \alpha |K(O)| + (1 - \alpha) |\varepsilon(O)| \]  

(25)

\(|K(O)|\) is the number of layers at orientation \(O\), which is normalized by dividing the maximum of layer numbers over all orientations. \(|\varepsilon(O)|\) is the geometric error at the orientation \(O\), which is also normalized by dividing the maximum geometric error over all orientations. The weighted ratio \(\alpha\) is set as 0.5.

We use the “Pig” model as an example to show the usage of metric profile to find the optimal building orientation. This “Pig” model’s height is 58mm. We uniformly sample the orientation space into 1000 directions (Fig. 8(a)). For each orientation, given the allowable maximum cusp height, we run our adaptive slicing algorithm, and obtain an optimal slicing plan and corresponding fabrication time and surface roughness. And the weighted objective energy is visualized in a Gaussian sphere as shown in Fig. 8(b,c), where the front-view and back-view of the Gaussian sphere are displayed with red and blue color represent large and small energy respectively. Three local best building orientations are shown in (d), (e), and (f).

5. Result

Our slicing pipeline is implemented in C++, built on the open source package LDI [38]. All tests are run by a PC with Intel(R) Core(TM) i5-3450 CPU @3.10GHz, 12GB RAM, and NVIDIA GeForce GT 640. The LDI sampling resolution is set as 2048. The heights of all the tested CAD models range from 17.8 mm to 62.4 mm. All the fabrication results are built using the SLA process [39]. We modified a Projet 1000 printer from 3D Systems to function with different layer thickness. The layer thickness is controlled by a Z-stage, and the practical thickness for a layer is from \(t_{\text{min}} = 50 \mu\text{m}(0.002\text{inch})\) to \(t_{\text{max}} = 150 \mu\text{m}(0.006\text{inch})\). The photo curing time of one single layer varies from 1 to 2 seconds respectively, while the overhead of transition between layers takes around 14 seconds.

Our implementation subdivides the metric profile into intervals with size \(b\). Intuitively, the value of \(b\) should not be greater than the minimum layer thickness \(t_{\text{min}}\) that can be printed (i.e., \(b \leq t_{\text{min}}\)), otherwise the algorithm cannot optimize the position of the layers with minimum thickness; as an approximation, it is also preferred that \(b\) should be as small as possible to minimize information loss due to discretization. To determine a practical value of \(b\), we have conducted an experiment on a “Hand” model with different \(b\) values and listed the results in Table 1.

Without surprise, the computational time increases with the number of intervals (i.e., decrease in the \(b\) value), but the number of layers required to achieve the same quality is also decreased due to better approximation. For a balance between quality and speed, \(b = 2 \mu\text{m}\) is selected in this paper.

5.1. Slicing Efficiency

Table 2 compares the slicing software time of our adaptive slicing (\(T_{\text{our}}\)) with other slicing methods, including uniform slicing with both coarsest (\(T_{\text{loc.}}\)) and finest (\(T_{\text{fin.}}\)) layer thickness, local greedy adaptive slicing (\(T_{\text{loc.}}\)) introduced in paper [3], and the global slicing using ultra fine resolution (\(T_{\text{ultra}}\)), i.e. our sampling resolution \((t = 12.5 \mu\text{m})\), which is much smaller than the finest layer thickness \((t = 50 \mu\text{m})\).

From the slicing results, our new pipeline consumes less than 1 second in total including sampling, constructing profile, generating and optimizing the slicing plan. In contrast, the direct slicing algorithms need around 5 seconds for coarsest layer thickness and 15 seconds for finest layer thickness. It is worth reminding that the slicing time for the local greedy adaptive slicing algorithm [3] is only one

![Figure 8](image-url)

Figure 8: (a) We uniformly sample the directions on a Gaussian sphere, and visualize the distribution of energy in a Gaussian sphere. Both the front view (b) and the back view (c) are displayed. The three best building orientations are shown in (d), (e), and (f).
Table 2: Slicing Efficiency Comparison

<table>
<thead>
<tr>
<th>Input</th>
<th>#Tri</th>
<th>#L</th>
<th>T_{our}</th>
<th>T_{co.}</th>
<th>T_{loc.}</th>
<th>T_{fine}</th>
<th>T_{ultra}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass1</td>
<td>30K</td>
<td>515</td>
<td>0.62</td>
<td>4.8</td>
<td>10.7</td>
<td>14.4</td>
<td>148.2</td>
</tr>
<tr>
<td>Glass2</td>
<td>30K</td>
<td>344</td>
<td>0.72</td>
<td>5.2</td>
<td>10.3</td>
<td>15.4</td>
<td>160.3</td>
</tr>
<tr>
<td>Knee</td>
<td>39K</td>
<td>625</td>
<td>0.58</td>
<td>10.3</td>
<td>19.1</td>
<td>30.8</td>
<td>328.7</td>
</tr>
<tr>
<td>Hand</td>
<td>84K</td>
<td>233</td>
<td>0.53</td>
<td>15.8</td>
<td>24.4</td>
<td>47.6</td>
<td>377.3</td>
</tr>
<tr>
<td>HearAid</td>
<td>33K</td>
<td>286</td>
<td>0.81</td>
<td>3.5</td>
<td>7.0</td>
<td>10.4</td>
<td>140.8</td>
</tr>
</tbody>
</table>

The time units are in seconds. #Tri is the number of triangles in each STL model. #L is the number of layers by our slicing algorithm. T_{our}, T_{co.}, T_{loc.}, T_{fine}, T_{ultra} represent slicing time of our algorithm, coarsest uniform, local greedy, finest uniform, ultra resolution slicing.

single evaluation of a slicing plan, but our adaptive slicing algorithm evaluates all the possible slicing plans with the ultra fine resolution and outputs the global optimal one. For comparison, we have also shown the needed time for slicing plan optimization that is directly computed on the CAD model using the ultra fine resolution, denoted as T_{ultra} in Table 2. On average, our algorithm could achieve 100 times faster than the direct slicing. The effect is more significant when a large number of slicing will be performed to optimize the fabrication orientation. In our orientation optimization experiment, it takes us less than 15 minutes to search among 1000 directions, while the slicing method directly computed on the CAD model needs more than 1 day.

5.2 Slicing Quality

In Table 3, we compare our method with other slicing algorithms, including slicing methods using uniform layer thickness and a greedy adaptive slicing method proposed in [3]. The error metric used in Table 3 is “Cusp Height”. The left part of Table 3 shows the number of layers in the results computed by different slicing algorithms. It also reflects the printing time as the printing time is almost proportional to the number of layers. The right part of Table 3 shows the geometry error computed by the maximum error among all layers, i.e. \( \max \{ \varepsilon_k \} \). Here, the cusp height is used as the error metric, and thus, this geometric error is also the maximum cusp height among all layers. Without any surprise, the coarsest slicing always gives the smallest number of layers but has the largest geometric error, and the finest one always has the smallest geometric error but gives the largest number of layers. They are used as the upper bound and the lower bound for comparing the adaptive slicing algorithms.

Both of our proposed and the greedy [3] adaptive slicing algorithm have set the allowable geometric error as the cusp height of \( \varepsilon = 65 \mu m \) for one layer. From the table, we can see that the two methods fall right inside the upper and the lower bound in terms of the layer number and geometric error. The results verify that both of the methods are effective. However, our algorithm can achieve a smaller number of layers under the same tolerance, and thus a smaller printing time (save up to 16% printing time). Moreover, due to the geometric complexity of the model and the fabrication constraints, the greedy algorithm fails to distribute the errors evenly on the layers. Hence their results are not optimal, and even cannot always satisfy the given tolerance for all the layers. In contrast, our algorithm has a global information for the planning, and our results can successfully satisfy the given tolerance in all cases. Last but not least, it is found that our adaptive slicing can get very competitive results with the finest slicing, but our results have much fewer number of layers and can save up to 49% printing time. It validates our proposed method is not only efficient but also promising.

5.3 Other Examples and Fabrication Results

We have applied our adaptive slicing algorithm using “Cusp Height” error metric on various models as shown in Fig. 9. These slicing results reveal an intuition that the layer thickness gets smaller at the height with increased geometry error density (i.e. with a larger value in the metric profile). This intuition results from the error constraint in each layer in eqn. 2, that is the error of each layer can not exceed the limit \( \varepsilon \). Hence, if the error density of a layer is large, then the layer thickness should be small so that the total integral error in the layer does not exceed the limit \( \varepsilon \).

Figure 10 displays the physical fabrication results using
Table 3: Slicing performance comparison with other slicing algorithms

<table>
<thead>
<tr>
<th>Input Model</th>
<th>Number of Layers</th>
<th>Geometric Error (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Height</td>
<td>#Tri</td>
</tr>
<tr>
<td>David</td>
<td>17.8</td>
<td>42K</td>
</tr>
<tr>
<td>HearAid</td>
<td>22.3</td>
<td>33K</td>
</tr>
<tr>
<td>Hand</td>
<td>23.8</td>
<td>84K</td>
</tr>
<tr>
<td>Laurana</td>
<td>37.4</td>
<td>51K</td>
</tr>
<tr>
<td>Glass2</td>
<td>39.0</td>
<td>30K</td>
</tr>
<tr>
<td>Glass1</td>
<td>55.1</td>
<td>30K</td>
</tr>
<tr>
<td>Pig</td>
<td>58.0</td>
<td>729K</td>
</tr>
<tr>
<td>Knee</td>
<td>62.4</td>
<td>39K</td>
</tr>
</tbody>
</table>

† We compare our method with the finest uniform slicing, greedy adaptive slicing [3], and coarsest uniform slicing, denoted as “Finest”, “Greedy” and “Coarsest” in the table. A set of models with various heights are sliced, and the results are compared in terms of the number of layers (the middle part of the table) and the geometric error (maximum layer’s integral error among all layers, unit:mm, the right part of the table). The height unit is mm, #Tri means the number of triangles in each STL model, and the geometric error uses the metric “Cusp Height”.

Figure 10: Fabrication results of hearing aid model by different slicing methods.

different slicing algorithms. From the microscope images (the bottom row of Fig. 10), we can see that the results from our slicing algorithm using “Cusp Height” error metric is very close to the one produced by the slicing with the finest layer thickness (the very left image in Fig. 10). Comparing with the results of the uniform slicing and local adaptive slicing, our result has a better surface finish, i.e. our result has less noticeable cusps and staircases. Figure 11 is another example to validate our slicing method. Similar with Fig. 10, the result using our method has fewer staircase defects comparing with other slicing algorithms like greedy and uniform slicing.

6. Conclusion and Discussion

This paper presents a novel adaptive slicing algorithm for the layer-based additive manufacturing. Traditional adaptive slicing algorithms suffer from long computation time or yield sub-optimal slicing result based on local geometry error. To generate the global optimal slicing plan efficiently, we introduce a novel algorithm based on a “metric profile”, which is a measure of geometry error distribution along a given building direction. The efficiency and effectiveness of our algorithm are enabled by three key ideas in our paper: 1) the new representation of metric profile provides us the global geometry deviation along the $z$ direction, rather than only geometry error on the slicing planes that are used in most traditional methods, 2) we efficiently construct the metric profile using a GPU-accelerated sampling approach, and 3) an optimization algorithm based on dynamic programming is proposed to find the global optimal slicing plan efficiently. Such advantages have been validated by comparing the computational time and fabrication quality with other slicing algorithms.
Our slicing method is a general adaptive slicing framework, and we have extended it to consider weighted features of a CAD model and to incorporate the commonly used surface quality metrics, such as cusp height, surface roughness, area deviation, and volume deviation. Our slicing algorithm can also serve as a building block for computing the optimal printing direction.

Our future works will be applying this framework to other geometric computation tasks that are required in the pre-fabrication pipeline of AM processes, and to speed up the process planning for AM applications like mass customization.

Acknowledgement

The work is partially supported by NSF grant number CMMI 1151191 and the Epstein Institute at USC.

Reference

[27] T.-H. Kwok, Y. Chen, C. C. L. Wang, Geometric analysis and computation using layered depth-normal images for three-
Appendix A. Approximation Errors Analysis

To determine a new slicing plane efficiently, the conventional slicing algorithm usually considers the normal of the intersection points by the previous slicing plane. In other words, it uses the normal to linearly approximate the surface between the two slicing planes, and uses the cusp height to determine the slicing error. When the layer thickness is small or the true surface is close to linear, such as the one shown in Fig. A.1(a), the cusp height is a good estimation of the approximation error during slicing and is widely used. However, such simplification can lead to inaccurate estimation when the surface is highly curved and a layer of fabrication becomes much thicker.

Before the analysis, the terminologies are defined first for clarification. The “true geometry” refers to the designed CAD model; “printed geometry” refers to the physical geometry created by the printer, shown as the solid blue portion in Fig. A.1.; “approximated geometry” indicates the estimated printed-geometry using the error metric before the physical printing, e.g., the geometry enclosed by the dashed line in Fig. A.1. Correspondingly, the “true slicing error” means the geometry discrepancy between the true geometry and the printed geometry; and the “approximated error” means the geometry discrepancy between the approximated geometry and the printed geometry.

The idea of our proposed metric profile is illustrated in Fig. A.1(b) using three smaller layers. The cusp heights are calculated separately in each small layer, and they are integrated as the error for the whole layer. In this case, the proposed integral error metric just recovers the cusp height of the thick layer as the big layer’s cusp height is the sum of three small layers’ ones.

Nevertheless, when the true surface is not linear, simply considering the normal of the previous plane results in a poor approximated surface and inaccurate error estimation. For example, if the bottom plane’s normal is used for the curve shown in Fig. A.1(c), the approximated surface is outlined by the dotted line. The true slicing error for this curve surface should be similar to (probably a bit smaller than) the one shown in Fig. A.1(a), because the true geometry in Fig. A.1(c) is a slightly concave shape of the one in Fig. A.1(a). Unfortunately, because the approximated geometry is different from the true geometry and the conventional method only takes a single normal vector, but the actual curved surface has many different normal vectors, the slicing error is computed as 0.86, which is much higher than the linear one 0.72.

In comparison, our error metric profile records the normal at each height, illustrated in Fig. A.1(d) using three small layers again. The cusp height of each layer is calculated using the normal in its own height. The sum of three small layer’s cusp height is 0.16 + 0.24 + 0.29 = 0.69, which is a bit smaller than 0.72 calculated from the similar linear curve. If we further divide the smaller layers, an even better approximation will be obtained. When the small layers are infinitesimal, this calculation converges to the error obtained by integrating the error metric profile.