Saliency-Preserving Slicing Optimization for Effective 3D Printing

Weiming Wang[1] Haiyuan Chao[2] Jing Tong[2] Zhouwang Yang[3] Xin Tong[4] Hang Li[3] Xiuping Liu[1] Ligang Liu[†3]

[1]Dalian University of Technology, China
[2]Hohai University, China
[3]University of Sciences and Technology of China, China
[4]Microsoft Research Asia, China

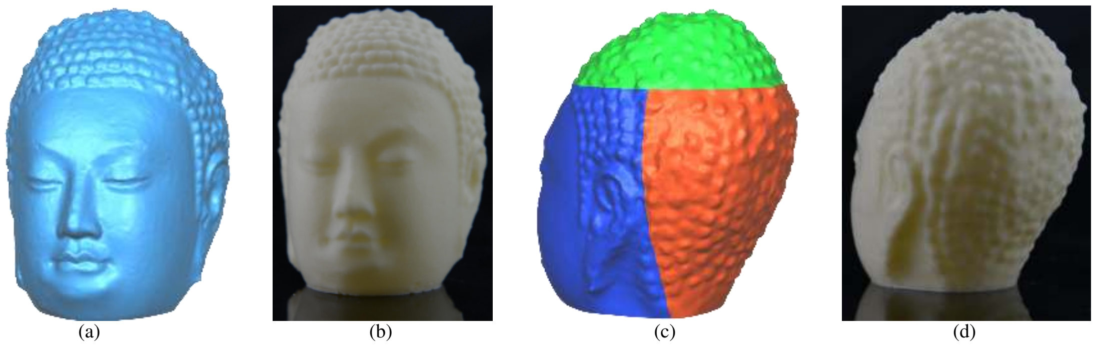

Figure 1: *Given the 3D buddha head model (a), the object printed using our slicing algorithm (b) has indistinguishable visual quality with the finest one printed by the highest resolution and saves 29.44% printing time. By applying the segmentation method, the segmented object (c) is printed which saves 39.11% printing time while also preserves the visual appearance (d). All of the savings are relative to the objects printed with finest resolution.*

Abstract

We present an adaptive slicing scheme for reducing manufacturing time of 3D printing system. Based on a new saliency based metric, our method optimizes the thicknesses of slicing layers to save the printing time and preserve the visual quality of printing results. We formulate the problem as a constrained ℓ_0 optimization and compute the slicing result via a two-step optimization scheme. To further reduce the printing time, we develop a saliency based segmentation scheme to partition an object into subparts and then optimize the slicing of each subpart separately. We validate our method with a large set of 3D shapes ranging from CAD models to scanned objects. Results show that our method saves 30-40% printing time and generates 3D objects that are visually similar to the ones printed with the finest resolution.

Keywords: *3D printing; Mesh saliency; Adaptive slicing; Segmentation; Visual quality*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computer Graphics—Computational Geometry and Object Modeling

1. Introduction

In recent years, 3D printing technique has received considerable attentions from graphics researchers due to its rela-

† lgliu@ustc.edu.cn

tively low cost and high capability for fabricating 3D objects with arbitrarily complex geometric shapes or topology [DSD-B06]. A set of techniques have been developed for manufacturing real 3D objects with various sizes, dynamics, appearances, as well as physical properties [SVB*12, LBRM12, P-WLSH13, WWY*13].

Despite these technical advances, the manufacturing speed of 3D printers is still very slow, which limits the usage of the 3D printers in many applications. Most Fused Deposition Modeling (FDM) 3D printers fabricate a 3D solid object by accumulating its volume layer by layer. To achieve the best quality result, an input object is sliced uniformly with the finest resolution of printer hardware and thus results in a large number of layers. As a result, a FDM printer always takes couple of hours to fabricate a small sized object with 10cm height.

A naive solution for improving the printing speed is to increase thickness of all layers. Although this method reduces the number of layers of input 3D shapes, the discontinuous boundary between the layers greatly decreases the result quality. Other methods slice input shapes adaptively by minimizing the local geometry error [PRD03]. Although these approaches work well for CAD models with sparse sharp features, they could not well handle the models with big salient regions that are popular in current 3D printing applications. Moreover, the local and heuristic optimization schemes adopted in these methods only achieve suboptimal results.

In this paper, we present an adaptive slicing method for reducing manufacturing time of 3D printers while preserving the visual quality of printing results. The key observation of our method is that human beings are more sensitive to geometric errors in the region with high visual saliency. We thus propose a saliency based geometric error metric for evaluating the visual quality of printing result. Based on this metric, our method slices input object with the least number of layers that can still preserve visual quality of the input after printing. We formulate this problem as a constrained ℓ_0 optimization and solve the thicknesses of slicing layers with two-step optimization. The first time optimization step is for eliminating redundant layers and the second visual optimization step is for minimizing the visual degradation of the printing result. We perform these two steps iteratively to reduce the number of layers until the visual quality degradation of the printing result achieves a user specified threshold.

To further reduce the printing time, we also present a saliency based segmentation method for partitioning input object into subparts, each of which is sliced independently with our adaptive slicing scheme. In final fabrication, each subpart is printed with its own slicing scheme and all subparts are automatically merged together to avoid apparent cutting seams. Figure 1 illustrates a result of our segmentation method.

We have tested our method with a large set of 3D objects including both CAD models and scanned 3D objects with rich surface details. We also evaluate the printing results generated by our method qualitatively and quantitatively. Results show that our saliency based metric works well for both kinds of input and the new optimization scheme generates better results than other existing local optimization scheme. Compared to the uniform slicing method with finest resolution, our method saves about 30-40% printing time and generates visually similar results.

The contributions of this paper are

- A visual saliency metric for guiding the adaptive slicing, which works well for both CAD models and 3D shapes with rich surface details.
- A constrained global ℓ_0 optimization for solving adaptive slicing of input object.
- A saliency based segmentation scheme for further reducing the printing time.

2. Related Work

3D printing A set of methods have been developed to extend the capability of current printing hardware for fabricating objects with large sizes [HFW11, LBRM12, VGB*14], various appearances [VWRKM13, LDPT13, LGX*13], and dynamics [BBJP12, ZXS*12, CCA*12, CTN*13, CLM*13]. Other methods optimize the current 3D printing system by enhancing results structure [SVB*12,ZPZ13] and stability [P-WLSH13], reducing material cost [WWY*13], and automating software pipeline [CLD*13]. Different from all these works, our work focuses on reducing the printing time in 3D printing.

Adaptive slicing Several adaptive slicing methods have been designed for manufacturing CAD models with variant layer thickness [PRD03], in which the result quality is measured by geometric errors (e.g. cusp height, chord length, and volumetric deviation, etc.). Most methods [DM94,JH95,TB98,HA13] optimize the slicing of input object with single geometric error defined over whole object surface. Mani et al. [MKD99] presented a region based adaptive slicing scheme that allow different surface regions have different geometric error. However, it is a difficult task for users to manually specify the acceptable geometric error over the object surface. Since these geometry-based methods do not take the surface saliency in consideration, they cannot guarantee the visual quality of the printing result. Moreover, all these methods optimize the thicknesses of slicing layers in a greedy or heuristic way and thus only generate suboptimal results. On the contrary, our method adapts the scheme in [HA13] for initialization and refines the results with a new constrained ℓ_0 optimization. Compared to geometric-based adaptive slicing solutions, our method well preserves the visual quality of printing results and saves more printing time.

Mesh saliency Inspired by image saliency works [IKN98, CZM*11], the mesh saliency metrics [SLM*14, LVJ05, C-SPF12,WSZL13] measure the importance of points or regions of a 3D surface mesh in a way similar to human visual perception. It has been proved that these perception-inspired metrics outperform pure geometry-based metrics for a set of mesh manipulation applications [LVJ05]. In our method, we pro-

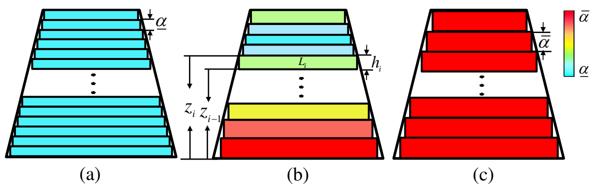

Figure 2: *Slicing a model into a set of layers with color coded by layer thickness. (a) uniform slicing of the input model with the minimal layer thickness $\underline{\alpha}$; (b) adaptive slicing of the input model; (c) uniform slicing of the input model with the maximal layer thickness $\overline{\alpha}$.*

pose a saliency-based metric for evaluating the visual quality of the printing result and use it to guide the slicing optimization and mesh segmentation [SLM*14].

3. Problem Formulation

Our Goal Given an input 3D model $\mathcal{G}$ with height H along the Z direction, a slicing scheme cuts $\mathcal{G}$ with a set of planes that are parallel to XY plane to generate a sliced mesh $\mathcal{M}$ along Z direction for 3D printing. As shown in Figure 2, slicing $\mathcal{G}$ uniformly with the minimal printable layer thickness $\underline{\alpha}$ specified by the 3D printer hardware results in the finest result $\underline{\mathcal{M}}$ with $N_{\underline{\alpha}} = H/\underline{\alpha}$ layers, which has the longest printing time. On the contrary, slicing $\mathcal{G}$ uniformly with the maximal printable layer thickness $\overline{\alpha}$ leads to the coarsest result with $N_{\overline{\alpha}} = H/\overline{\alpha}$ layers, which takes the shortest printing time. We denote the layer thickness vectors of the finest and the coarsest results as $\underline{\mathbf{h}}$ and $\overline{\mathbf{h}}$ respectively.

As shown in Figure 2(b), our adaptive slicing method generates a sliced mesh with N layers $\{L_1, L_2, \cdots, L_N\}$, where each layer thickness $h_i \in [\underline{\alpha}, \overline{\alpha}]$ and $\sum_{i=1}^{N} h_i = H$. Our goal is to slice the input model with optimal layer thickness $\mathbf{h} = (h_1, h_2, \cdots, h_N)$ so that the result takes as short printing time as possible and preserves the visual quality.

Printing Time Although there are many factors that would affect the printing time of a sliced mesh $\mathcal{M}$ [AAD98], we found that the printing time of a layer is almost constant no matter how layer thickness and cutting area vary because the sum of the travel moving time and retracting time is larger than wire squeezing time in most cases. Furthermore, the printer hardware related time cannot be optimized by us and the print path optimization algorithm is already very mature, so the major factor affects the printing time can be optimized is the number of layers. As a result, we determine the printing time $T(\mathcal{M})$ of a sliced mesh $\mathcal{M}$ by the number of its layers $T(\mathcal{M}) = N$ under the slicing orientation is fixed which can be optimized in Section 6.

Different from existing solutions that directly reduce the number of layers with heuristic layer merging or splitting scheme, we set the number of layers of a sliced mesh $\mathcal{M}$ as the one of the finest result $N_{\underline{\alpha}} = H/\underline{\alpha}$ and degenerate the thicknesses of redundant layers as zero and optimize the slicing positions of the rest layers. In particular, $h_i \in [\underline{\alpha}, \overline{\alpha}]$ for the valid layers and $h_i = 0$ for the degenerated layers. As a result, the printing time of the sliced mesh $T(\mathcal{M})$ can be defined by the ℓ_0-norm (the number of non-zero elements) of $\mathbf{h}$, i.e., $||\mathbf{h}||_0$, where $\mathbf{h} = (h_1, h_2, \cdots, h_N)$.

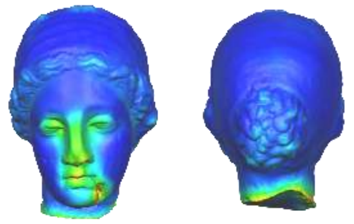

Figure 3: *Saliency maps of the Venus head model. Regions around the face with larger saliency are more visually important than the other regions.*

Visual Quality of Printing Result To evaluate the visual quality of a sliced mesh, we first measure the visual importance of the surface points over the mesh and then define the saliency based metric for measuring the visual quality of the printing result. Finally, the visual degradation of sliced mesh generated by our method is computed by the difference between its visual quality and the one of the finest result.

Specifically, we apply the method in [SLM*14] to compute the saliency S of original input model $\mathcal{G}$. As shown in Figure 3(left), the result is a scalar measure of visual importance $\in [0.0, 1.0]$ for each point on $\mathcal{G}$, where 0.0 represents the points with the lowest visual importance and 1.0 refers to the highest visual importance.

After that, we define the saliency S_i of i-th layer L_i in the result sliced mesh $\mathcal{M}$ of $\mathcal{G}$ as the largest saliency value of all surface points in the layer. Based on the mesh saliency, we measure the visual quality of the i-th layer L_i by a saliency based metric (SM)

$$SM(L_i) = S_i \cdot h_i \cdot \cos\theta_i, \tag{1}$$

where $h_i \cdot \cos\theta_i$ is the maximal cusp height of the layer L_i. $\theta_i = \min_{p \in L_i} \theta(p)$ is the minimum of the angles $\theta(p)$ computed on all surface points in the layer, in which $\theta(p)$ is the angle between the normal of a surface point p and the Z direction. We sum the visual quality metric of all layers to compute the visual quality of the sliced mesh

$$SM(\mathcal{M}) = SM(\mathbf{h}) = \sum_{i=1}^{N} SM(L_i). \tag{2}$$

Finally, we regard the finest result $\underline{\mathcal{M}}$ as the ground truth and compute the visual quality degradation of our result mesh $\mathcal{M}$ as the SM difference of two meshes

$$E(\mathbf{h}) = SM(\mathbf{h}) - SM(\underline{\mathbf{h}}). \tag{3}$$

Note that the visual quality degradation of a slicing result $E(\mathbf{h})$ is between 0.0 and $E(\overline{\mathbf{h}})$.

Slicing Optimization Based on the printing time and the

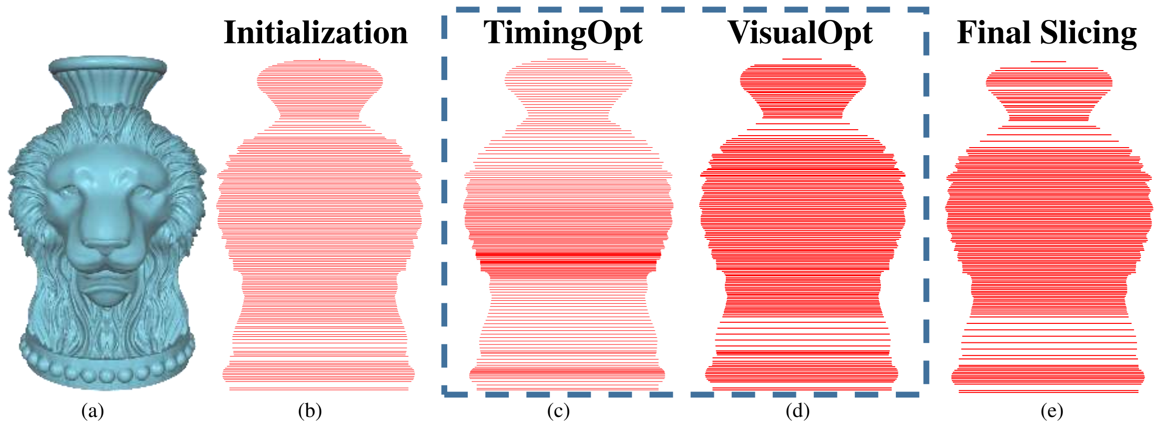

Figure 4: *Overview of our adaptive slicing algorithm. Given an input model (a), our algorithm initializes the slicing result (b). Then our algorithm iteratively executes timing optimization (c) and visual optimization (d) until the visual degradation of the sliced mesh exceeds the user specified threshold. The final slicing result is (e).*

visual quality metric defined above, we compute an optimal slicing solution **h** via a constrained ℓ_0-norm optimization

$$\begin{aligned} \min_{\mathbf{h}} \quad & \|\mathbf{h}\|_0 \\ \text{s.t.} \quad & h_i \in \{0\} \cup [\underline{\alpha}, \overline{\alpha}], i = 1, \cdots, N \\ & \textstyle\sum_{i=1}^N h_i = H, \\ & E(\mathbf{h}) < \varepsilon, \end{aligned} \tag{4}$$

where $\|\mathbf{h}\|_0$ determines the printing time, and $E(\mathbf{h})$ computes visual quality degradation of the printing result as Equation 3. $\varepsilon \in [0.0, E(\underline{\mathbf{h}})]$ is a threshold specified by the user to control the visual quality of the printing result.

4. Adaptive Slicing (AdapSlice)

The constrained optimization problem defined above 4 is NP hard because the first constraint is a semi-continuous interval constraint so that formulation 4 is a mixed integer programming problem which is NP-hard. We solve the problem with a two-step optimization as shown in Figure 4. After initialization, we first apply a timing optimization to degenerate the redundant layers. Then a visual optimization step is executed to adjust the layer thickness and minimize the visual degradation of the sliced mesh. We repeat these two steps iteratively until the SM difference $E(\mathbf{h})$ exceeds the given threshold ε.

Initialization Starting from $\mathbf{h} = \underline{\mathbf{h}}$, we adopt the greedy scheme in [HA13] to merge the layers for initialization. To this end, we first compute the merge cost $E(L_i \cup L_{i+1})$ for each layer L_i as the SM difference between the original two layers L_i and L_{i+1} and the merged one. If the total thickness of L_i and L_{i+1} exceeds the maximal printable layer thickness $h_i + h_{i+1} > \overline{\alpha}$, we assign a large enough merge cost to this layer so that it will not be merged. After that, we merge the layers according to their merge costs in the ascent order until the total merge costs exceeds the user given threshold ε.

Timing Optimization After initialization, we perform timing optimization (TimeOpt as short) to degenerate some layers to save printing time. To this end, we relax semi-continuous interval constraint in Equation 4 by a continuous one

$$h_i \in [0, \overline{\alpha}], \tag{5}$$

where $h_i = 0$ means that the slicing layer L_i is degenerated. Similar to the method in [WWY*13], we solve the ℓ_0-norm optimization in Equation 4 by reformulate it as a weighted ℓ_1 optimization

$$\begin{aligned} \min_{\mathbf{h}} \quad & \|\mathbf{Wh}\|_1 \\ \text{s.t.} \quad & h_i \in [0, \overline{\alpha}], i = 1, \cdots, N \\ & \textstyle\sum_{i=1}^N h_i = H, \\ & E(\mathbf{h}) < \varepsilon, \end{aligned} \tag{6}$$

where $\mathbf{W}$ is a diagonal matrix and $\mathbf{w} = \mathrm{diag}(\mathbf{W})$ is a favorable weight vector designed for counteracting the influence of the slicing thickness magnitude on the ℓ_1-norm objective. Based on the solution **h** from the initialization or the visual optimization, a desired weight vector can be constructed by

$$w_i = \frac{1}{\eta + (h_i' - \underline{\alpha})}, \tag{7}$$

where a small number $\eta = 10^{-8}$ is set to provide numerical stability. We solve this timing optimization with the interior-point algorithm described in [NW06].

After this optimization, the thicknesses of some layers are less than $\underline{\alpha}$ and thus can be removed. For this purpose, we collect all redundant layers whose thicknesses are below $\underline{\alpha}$ and compute their merging costs

$$E_{L_i} = min(E(L_{i+1} \cup L_i), E(L_{i-1} \cup L_i)). \tag{8}$$

We also record the corresponding merging neighbor layer of each redundant layer L_i as L_{i+1} if $E(L_{i+1} \cup L_i) < E(L_{i-1} \cup L_i)$ or L_{i-1} otherwise. After that, we degenerate these layers according to the ascendent order of their merge costs until the SM difference of the merged slicing mesh exceeds the user given threshold ε. In each step, we set the thickness of current redundant layer to zero and update the thickness of the corresponding merging neighbor layer as sum of two layers.

Visual Optimization After timing optimization, we obtain the updated thickness vector $\mathbf{h}$. In visual optimization (VisualOpt as short) step, we fix degenerated layers unchanged and refine the thickness of non-zero layers to minimize the total SM difference of sliced meshes. To this end, we have

$$\begin{aligned} \min_{\mathbf{h}'} \quad & E(\mathbf{h}') \\ \text{s.t.} \quad & h'_i \in [\underline{\alpha}, \overline{\alpha}], i = 1, \cdots, N' \\ & \textstyle\sum_{i=1}^{N'} h'_i = H, \end{aligned} \tag{9}$$

where $\mathbf{h}'$ is the thickness vector of all non-zero layers and N' is the number of non-zero layers. We also apply the interior-point algorithm [NW06] to solve the visual optimization.

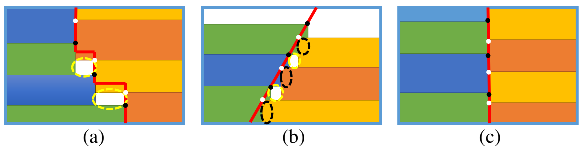

Figure 5: *Voids (marked in yellow dotted circles) and overlaps (marked in black dotted circles) exist if the cutting boundary is not vertical and the two subparts are sliced independently, as shown in (a) and (b). Thus we allow only vertical planes to segment the object in order to avoid them (c).*

5. Saliency based segmentation

Given a 3D model, the adaptive slicing algorithm 4 can be applied on the whole model and obtain an adaptive slicing solution. However, for some models, different regions at the same slice might have much different saliency. For example, the back region of the Venus head model, as shown in Figure 3(right), is less salient than the front face part of the model (Figure 3(left)). Thus it is not efficient to slice the back region with the same high resolution as the face part. To further reduce the printing time, we segment the object into several subparts so that each subpart can be sliced separately with our slicing optimization method.

Existing saliency based segmentation methods (e.g. [SLM*14]) could segment the input model into regions with different saliency. However, the algorithm is not designed for 3D printing. We thus present a saliency based segmentation algorithm by considering the geometric constrains posed by 3D printing hardware.

Printing Hardware Constraints Instead of printing subparts individually and then assembling and gluing them into an object, we want to print all subparts of an input model as a whole (as shown in Figure 1(d) and Figure 6(c)). Since the subparts with a same cutting boundary are sliced independently, their slicing plane positions might not align along a cutting boundary, as shown in Figure 5(a) and (b). As a result, some voids or overlaps may occur along the cutting boundaries during printing. We have found that the overlaps may damage the printer nozzle seriously. In order to avoid these artifacts, we use either horizontal or vertical cutting planes to segment the input model in our solution.

Although the horizontal cutting boundaries can be well printed (shown in Figure 5(c)), the vertical cutting plane still leads to some visual artifacts (shown in Figure 6(c)) when the object is printed with a FDM printer. Unfortunately, these artifacts could not be totally avoided due to the limitations of current FDM printing hardware and physical properties of printing materials(we have tested both PolyLactic Acid (PLA) and Acrylonitrile Butadiene Styrene (ABS) materials). Therefore, our segmentation method tries to reduce the number of vertical cutting boundaries as much as possible in order to minimize visual artifacts in the printing result.

Algorithm Overview With the constraints described above, our method segments the input model progressively. As shown in Algorithm 1, we first find an optimal vertical plane that can segment the input model into two parts. If the reduced printing time of the two subparts is less than a user specified threshold τ (0.85 in our current implementation), our method stops to segment the input model. Otherwise, we cut the input model with several horizontal planes again and then merge subparts between the two neighboring horizontal planes and thus remove the vertical cutting boundary between two subparts. After that, we execute the adaptive slicing scheme for result subparts again. This algorithm can be executed recursively for each result subpart when necessary. In our implementation, we execute this segmentation algorithm only once for all the input models shown in the paper.

Algorithm 1 SegOpt

Input: A 3D model $\mathcal{G}$ and $T(\mathcal{M})$ obtained by AdapSlice
Output: Its segmentation and the slicing results

```
1: Find an optimal vertical cutting plane which partitions $\mathcal{G}$
   into two subparts $\mathcal{G}_1$ and $\mathcal{G}_2$
2: Call AdapSlice for $\mathcal{G}_1$ and $\mathcal{G}_2$ separately to obtain $\mathcal{M}_1$,
   $\mathcal{M}_2$, $T(\mathcal{M}_1)$ and $T(\mathcal{M}_2)$
if ( $(T(\mathcal{M}_1)+T(\mathcal{M}_2))/T(\mathcal{M}) < \tau$ ) {
    3: Horizontal Segmentation and Merging;
    4: Call SegOpt for each subpart;
  }
```

Vertical Cutting Plane Optimization To find the optimal vertical cutting plane for segmentation, we parameterize the cutting plane with a 2D point in the $X-Y$ plane that passes and the 1D distance of the origin point to the plane. We then generate all possible cutting planes by uniformly sample parameter space (100×100 2D points on a unit circle in $X-Y$ plane and 50 1D distance samples on the diagonal of the bounding box of the model). For each cutting plane, we trace its intersection seam C with the input model surface and compute the cost function of C as

$$f(C) = \int_C (Q + \lambda_1 S)dC - \lambda_2 D, \tag{10}$$

where Q is the mean curvature of each seam point that is used to encourage the cutting seam C to locate on invisible regions [LBRM12]. S is the visual importance of each seam point that is used to constrain the seam to go through the non-salient regions. D measures the difference of the averaged visual importance of two subparts partitioned by C. Large D leads to less overall printing time as the subparts with small averaged visual importance may be sliced thicker and thus take less printing time after segmentation. λ_1 and λ_2 are the weights ($\lambda_1 = 0.3$ and $\lambda_2 = 0.4$). After computation, we choose the cutting plane with the minimal $f(C)$ as the optimal vertical cutting plane.

Horizontal Segmentation and Merging In order to reduce the artifacts on the vertical cutting seams, we tries to divide the two subparts with the horizontal planes and merge two subparts in each horizontal partition so that the vertical cutting boundary between the two subparts can be removed. To this end, we find a set of critical points [ZC95] along the z-direction for each subpart (see Figure 6(b)), at which the neighboring layer thickness abruptly changes (> 0.05 in our implementation). These critical points split the object into a set of intervals along the Z direction. If the average SMs of two subparts within an interval are similar (< 0.03 in our implementation), we segment the object with the two horizontal planes of this interval and then merge the two subparts within the interval and remove the vertical cut between them (as shown in Figure 6 (c)). By merging the subparts with similar SM errors, our method reduces the artifacts caused by the vertical segmentation and preserves the printing time of the input model.

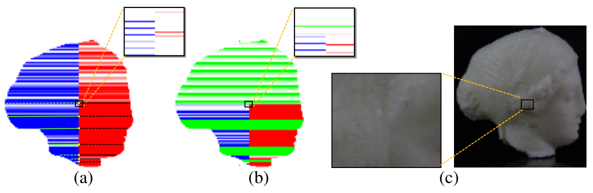

Figure 6: *The segmentation of the Venus head model. (a) and (b) show the steps of SegOpt. (a): the model is split into two subparts by a vertical plane and each subpart is sliced independently. The critical points of each subparts partition the model into multiple layer intervals. (b): some layer intervals are merged (shown in green) and each subpart is independently sliced. (c) shows the printed object. The regions in black rectangles are zoomed in and the slicing and artifact can be seen clearly.*

6. Model Orientation

Both of our slicing optimization and segmentation algorithm assume that the orientation of the input model is specified by user. If this is not the case, we adapt the method in [HBA13] to compute the object orientation that can save the printing time and preserve the printing quality. In particular, we follow the normal clustering scheme in [HBA13] to find three orthogonal candidate directions. Then we choose orientation to follow the candidate direction with the biggest orientation score that is computed by

$$O_S(\mathcal{M}) = a_1.Time_S(\mathcal{M}) + a_2.Vol_S(\mathcal{M}) + a_3.Stas_S(\mathcal{M}), \tag{11}$$

where $Time_S$, Vol_S and $Stas_S$ are printing time score, volumetric error score, and printing stability score respectively. a_1, a_2 and a_3 are their weights (0.3 in our implementation). The printing stability score is defined by $Stas_S(\mathcal{M}) = \frac{A_T(\mathcal{M})}{A_\mathcal{G}}$ [XWL*97], where $A_T(.)$ indicates the contact area between the object and the printing plate, $A_\mathcal{G}$ is the surface area of $\mathcal{G}$. The printing time score is defined

$$Time_S(\mathcal{M}) = \frac{MAX_T - T_i}{MAX_T - MIN_T}, \tag{12}$$

where T_i is the printing time of the object in current orientation. MAX_T and MIN_T are the maximal and the minimal printing time for objects in all three orientations. Different from [XWL*97] that uses the number of slicing layers of the input object as the printing time, we compute the printing time for both the input object and the supporting parts. For this purpose, we send the sliced mesh generated by our method to Cura Engine [Cur13] to generate the printing path for both input object and supporting part. We then compute the printing time based on the printing speed of the 3D printer.

The new volumetric error score Vol_S is computed by

$$Vol_S(\mathcal{M}) = 1 - \frac{|\sum_{k=1}^{N} h_k.P_k - V_\mathcal{G}|}{V_\mathcal{G}}, \tag{13}$$

where P_i is the perimeter of the ith layer, $V_\mathcal{G}$ is the volume of the input model $\mathcal{G}$.

Figure 7 shows an example of best slicing orientation selection. The corresponding orientation scores of three candidate directions shown in Figure 7 (b - d) are 0.5503, 0.2972 and 0.6927 respectively. As a result, the orientation in Figure 7 (d) is chosen as the best slicing orientation.

7. Results

Implementation Details We have implemented our algorithm in C++ on a PC with Intel(R) Core(Tm) i7-3770K CPU @ 3.50 GHz and 8 GB memory. The results are fabricated by a MakerBot Replicator^{TM} 2X, which is a FDM 3D printer with tray size $225\text{mm} \times 145\text{mm} \times 150\text{mm}$ [Mak12]. The printable layer thickness of the printer ranges from $\underline{\alpha} = 0.1\text{mm}$ to $\overline{\alpha} = 0.4\text{mm}$. In all experiments, we scale the input model uniformly so that its height is 80mm. After optimization, the number of layers of the result sliced mesh varies from $N_{\overline{\alpha}} = 200$ to $N_{\underline{\alpha}} = 800$.

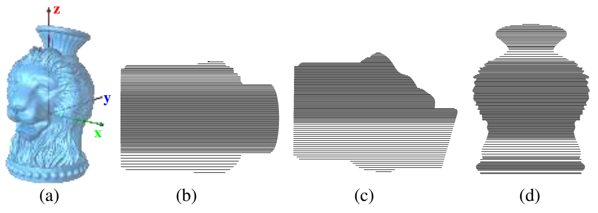

Figure 7: *Slicing orientation selection. Three orthogonal orientations are determined by [Hildebrand et al. 2013] (a), and the model is sliced along x, y and z orientations which are shown in (b), (c) and (d), respectively.*

All objects shown in this paper are printed as a shell with thickness of 1.2 mm. If a model needs to be printed as solid, we follow the method in [Sab96] to decompose the model into interior part and exterior part. The exterior part is sliced by our adaptive slicing algorithm, while the interior part is sliced by the coarsest resolution $\overline{\alpha}$. Finally, we use open source slicing engine Cura [Cur13] to convert the result sliced mesh into the machine code (G-code) so that it can be fabricated by the 3D printer. We have managed to customize the Cura engine to generate modified G-code so that we can generate layers with different thickness and print the subparts of an input model as a whole. The slicing paths of each part of the model are generated by 5 and all these paths must be assembled together to form one G-code so that the printers can print them together. Firstly, the slicing paths of each part are generated independently; then, the slicing paths of all parts are sorted according to ascending order of the slicing height; finally, the slicing paths are written into the file one by one according to the order. The generated G-code has been tested on FDM printers successfully (see Figure 1 (d) and Figure 6 (c)). It is worth noting that the difference between two adjacent slicing height cannot exceed the maximal printable layer thickness. Supporting structures are used to keep the balance of the object during the printing and it is no longer useful after the object is printed completely, so the printing accuracy of the supporting structures does not affect the accuracy of the whole object. Therefore, the slicing thickness and slicing positions of the supporting structures are not need to be optimized which can be the same as the model.

Parameter The parameter $\varepsilon \in [0, E(\overline{\mathbf{h}})]$ used in our algorithm balances between the printing time and the visual quality of the printed object. Denote $\varepsilon = \rho E(\overline{\mathbf{h}})$. We allow the user to adjust $\rho \in [0,1]$ to control the result. Smaller value of ρ leads to results with better visual quality and longer printing time (more layers), while larger value of ρ leads to results with worse visual quality and shorter printing time (less layers), as shown in Figure 9. To guarantee the visual quality of printed objects, we prefer small value of ρ (we set $\rho = 0.05$ in our experiments).

Convergence Our algorithm stops when the SM difference of the VisualOpt exceeds the given tolerance. In general, it takes only 1-2 iterations to converge in AdapSlice and each iteration needs about 100 seconds to reach the local minimal.

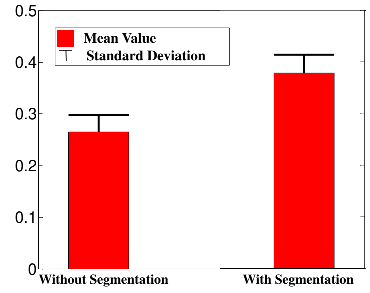

Figure 8: *The statistics of averaged saving time and variances on testing SegOpt on a dataset of 267 models. 80 models were not segmented as the partition does not lead to significant time saving. The others were segmented into multiple subparts, which save more time in the printing.*

Figure 10 only shows the changes of the number of layers and the SM difference of the initialization and first two iterations, since the SM difference of the VisualOpt of the third iteration exceeds the user given SM difference tolerance. The number of layers is guaranteed to decrease during the iterations (Figure 10 (left)). Each iteration is shown in the skyblue dotted box. The green line shown in the Figure 10 (right) is the SM difference tolerance given by user. The blue dots in the Figure 10 denote the number of layers and the SM difference after corresponding operations, respectively. At the initialization, the SM difference will exceed the given tolerance when a new layer is removed, so the heuristic merging in the initialization cannot remove more layers (that is to say, it cannot reduce the printing time further) under the current given SM difference tolerance. However, the proposed AdapSlice can reduce the total SM difference further (See Figure 10 (right)) because TimingOpt can adjust the slicing positions of the layers during the optimization.

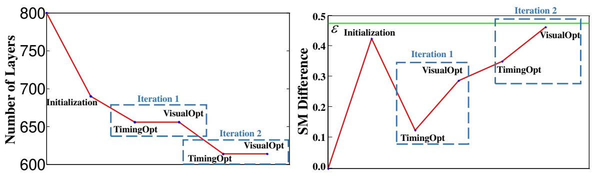

Figure 10: *The changes of the number of layers (left) and SM difference (right) during the initialization and first two iterations in SliceOpt (for the model in Figure 4 (a)).*

Algorithm Performance We have tested our slice optimization algorithm and segmentation algorithm with 267 3D models randomly selected from the SHREC database [SMKF04] and the Princeton database [LGB*11]. Among all tested models, 80 3D models are not partitioned because the segmentation does not lead to significant printing time saving. All the others are partitioned into three or more subparts after segmentation. Figure 8 illustrates the statistics of the reduced printing time of all results generated by our method.

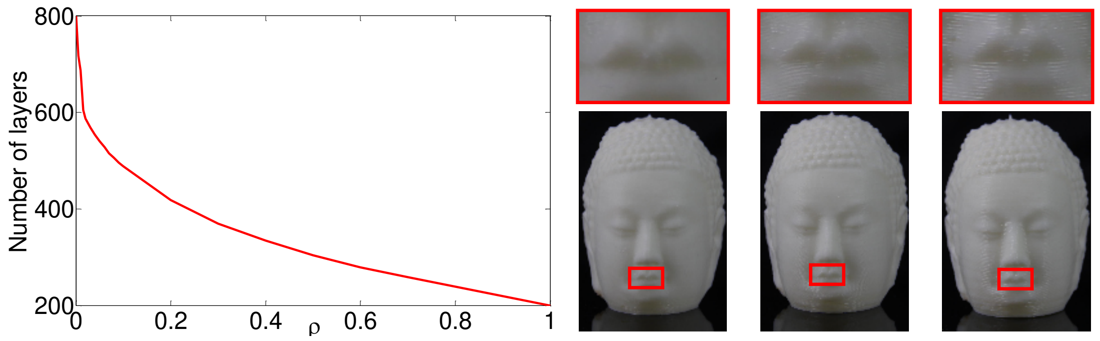

Figure 9: *The left shows the relationship between the parameter* ρ *(x-axis) and the number of layers (y-axis) generated by SliceOpt (for the model in Figure 1(a)). The right shows the three objects printed with different* ρ *(which are 0.05, 0.2 and 0.5 from left to right).*

Compared to the finest result, the results generated by slicing optimization save 26.78% printing time in average compared to the finest results. For results generated by both slicing optimization and segmentation, they save 34.23% printing time in average compared to the finest results. Figure 1 (c) shows a model which is partitioned into 3 subparts using SegOpt and the photo of the printed object is shown in Figure 1 (d). The models with large saliency variations over the model (like the Venus head model in Figure 3) can gain more benefits from our segmentation and slicing methods.

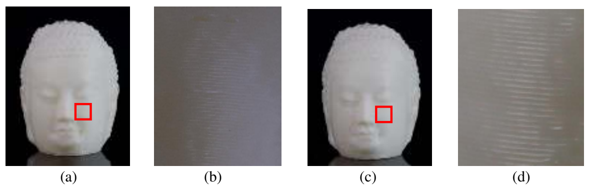

Figure 11: *The comparison of the visual quality between our method (a) and [HA13] (c). Figure (b) and (d) show the close-ups of the red marked regions.*

Method Validation We evaluate the printing time and the quality of the results generated by our method and other methods for six 3D models (see Figure 11, 12 and Figure 14). Here are the descriptions of all methods:

GroundTruth: the ground truth slicing with the finest resolution $\underline{\alpha} = 0.1\text{mm}$.

OurResult: the slicing result produced by our AdapSlice algorithm. The segmentation algorithm is not executed before slice optimziation for a fair comparison.

Random: the slicing result generated by randomly permuting the layers of the result generated by our method.

Uniform: uniform slicing result with the same layer thickness so that the printing time almost the same as OurResult.

DM94: the slicing result produced by the method in [DM94].

Sab96: the slicing result produced by the method in [Sab96].

HMS: the heuristic merging guided by our saliency metric.

CEO: geometric metric together with l_0 optimization.

HA13: the heuristic merging with geometric error defined by the triangle area of the projected slicers in [HA13].

Given the same number of layers (i.e. the same printing time), we compare the visual quality of the Buddha model between OurResult and HA13, see Figure 11. HA13 is a heuristic method and dose not take into account the visual saliency of the model which is more sensitive to human eyes. Therefore, HA13 can only achieve suboptimal results. On the contrary, our method adaptively slices the 3D object with a visual saliency based error metric. With a global l_0 optimization, our method can achieve better slicing results. In Figure 11, the region bounded by the red rectangle is salient by human eyes, but the projection area of the adjacent layers is very small. Therefore, our method can slice this region with finest resolution (see Figure 11 (b)), while HA13 slices it with thick thickness (see Figure 11 (d)).

We then compare OurResult with HMS and CEO to prove that the l_0 optimization cannot be replaced by heuristic merging strategy and saliency metric cannot be replaced by other geometric metric (such as cusp height). Table 1 lists the relative SM difference under the same printing time and the printing time under the same relative SM difference of OurResult, HMS and CEO, respectively. Here the relative SM difference (to GroundTruth) is defined by $RSMD(\mathbf{h}) = \frac{E(\mathbf{h})}{E(\bar{\mathbf{h}})}$, where $E(\mathbf{h})$ is the SM difference as defined in 3. Statistical result clearly shows that our method is the best one. Figure 12 (b), (g) and (h) show the visual effect of printed objects generated by OurResult, HMS and CEO, respectively. From the figure we can see that, the visual equality bounded by the red rectangle of Figure 12 (g) and (h) have obvious flaws, while Figure 12 (b) is perfect. Although HMS uses the saliency metric, it merges the adjacent layers heuristically. Therefore, the slicing positions of their method are suboptimimal and some layers may appear obvious flaws, see Figure 12 (g). The cusp metric is used in CEO which focuses on the regions with high curvature but not the high salient regions. Although CEO uses l_0 optimization, it still slices the top of the Buddha with finest resolution while slicing the face with thick thickness. l_0 optimization can help CEO optimize the slicing positions and the

Model	RSMD						RSMD	Printing Time (m)					
	OR	UT	C1	C2	HMS	CEO		OR	UT	C1	C2	HMS	CEO
Buddha	0.0068	0.1250	0.1516	0.1556	0.0781	0.1487	0.1880	169	207	232	238	201	248
Rabbit	0.0075	0.1299	0.1083	0.1111	0.0965	0.1037	0.1411	62	88	92	96	79	97
Squirrel	0.0031	0.1255	0.1196	0.0971	0.0802	0.1145	0.0996	139	178	175	170	169	180
Laurana	0.0038	0.1251	0.1520	0.0986	0.0921	0.1029	0.1093	149	198	202	193	187	199
Lion	0.0485	0.1252	0.1494	0.1333	0.1102	0.1265	0.1063	165	199	201	214	191	219
CleanHead	0.0028	0.1249	0.1948	0.1214	0.0387	0.1089	0.1276	87	115	132	127	108	142

Table 1: *Comparison of the relative SM difference under the same printing time (the left part of the table) and printing time under the same relative SM difference (the right part of the table) of different methods. OR, UT, C1, and C2 refer to the methods of OurResult, Uniform, DM94, and Sab96, respectively.*

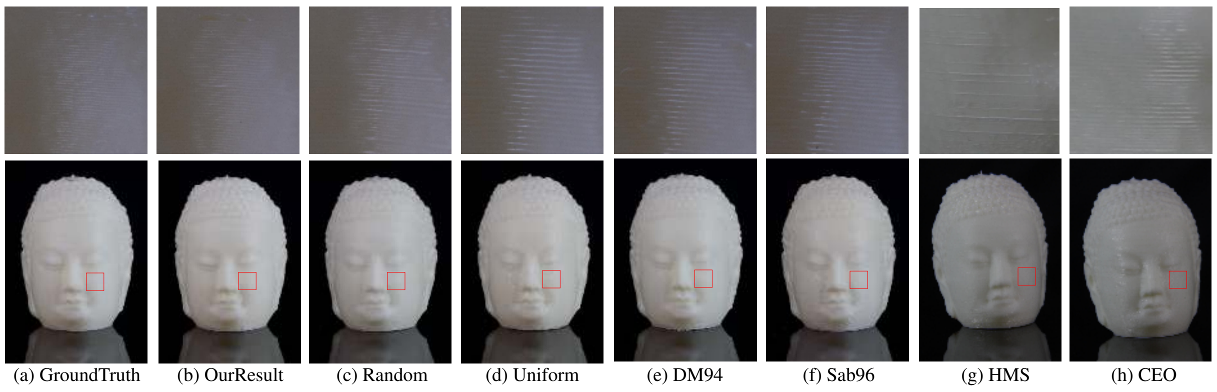

Figure 12: *Comparison of different slicing results for the Buddha head model. The first row shows the close-up photos of the regions in red and the second row shows the photos of printed objects.*

number of layers, but it cannot slice the high salient regions with finest resolution. Given the fixed number of layers (i.e., the same printing time), the finest layers obtained by CEO are located at the top of head, while the thick layers are located at face. Our method can slice the face of the Buddha with finest resolution according to the saliency metric and l_0 optimization. That is why the visual artifacts shown in the insets of the CEO are bigger than OurResult, see Figure 12 (h). In conclusion, our saliency metric and l_0 optimization cannot be separated and replaced.

Furthermore, we evaluate the visual quality of the results generated by different methods (OurResult, Random, Uniform, DM94 and Sab96) under the same printing time. The second to sixth columns of the Table 1 list the relative SM differences of the results generated by different methods. Note that the results generated by our method have the best visual quality except the GroundTruth. Figure 12 shows the photos of different slicing results. OurResult can achieve the finest slicing layers at high salient regions while DM94 and Sab96 focus on the regions with high curvature. Note that the results generated by our method preserve the details of the salient eye part of the head as the GroundTruth while the other methods produce coarser layers in the same region. From the close-up photos, visible artifacts can be seen in the results generated by Random, Uniform, DM94 and Sab96 methods. On the contrary, OurResult looks no apparent difference from the GroundTruth. As a result, the printed objects produced by our method have comparative visual quality with the ground truth and are much better than the other methods.

We also evaluate the printing time of the results that are generated by different methods (OurResult, Uniform, DM94 and Sab96) and have the same relative SM difference. The seventh column of the Table 1 lists the relative SM differences of different models. The last four columns of the Table 1 list the printing time of OurResult, Uniform, DM94 and Sab96. Since DM94 and Sab96 slice the input model based on cusp height metric only, they need to generate a large number of layers to preserve the visual quality of the printing result. On the contrary, our method adaptively slices the input model guided by the SM of the model. As a result, the printing time of OurResult is the smallest compared to other results.

More Results Figure 13 shows an example of slicing a sphere model. As the saliency is constant on the sphere model, the SM (1) is approximately equal to the cusp measurement used in [DM94]. More slicing results generated by our algorithm for CAD models are shown in the supplementary material. Figure 14 shows the photos of five printed objected whose slicing are generated using AdapSlice. Compared to the finest result, the results generated by our method take 26% - 34% less printing time. We also test AdapSlice on CAD models and its performance is similar to the traditional adaptive slicing method [DM94].

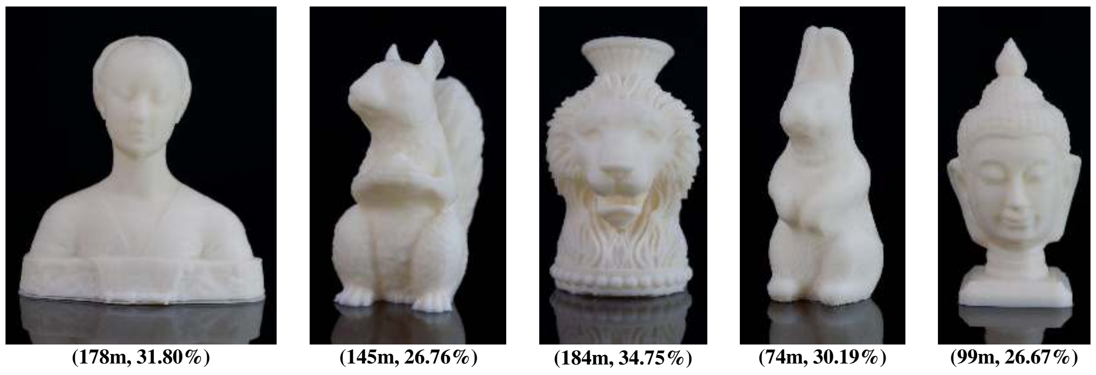

Figure 14: *Photos of the printed objects using AdapSlice. Numbers in brackets below each photo denote its printing time (in minutes) and its ratio of time saving according to the ground truth. As shown in the user study, there is no apparent visual difference between these objects and their corresponding ground truth.*

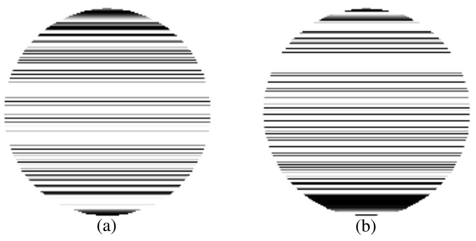

Figure 13: *Slicing results on a sphere model using different methods. (a) The result produced by the method of [Dolenc and Makela 1994]; (b) The result produced by AdapSlice.*

User Study We also conduct an user study to further evaluate the visual quality of the results generated by our method.

We choose 6 representative models from the dataset. For each model, we generate 6 slicing results by applying the 6 methods mentioned above and manufacture them using the FDM printer. Thus we have 6 groups of printed objects, where each group consists of 6 printed objects for the same model (see details in the supplementary material and the accompanying video). Each subject is asked to select the top 3 objects with good visual quality from each group and rank the 3 objects. In order to make the study more reliable, we create another group (named as JudgeG) of 6 printed objects, where each of them was created with a uniform layer thickness of $(1-\lambda)\underline{\alpha}+\lambda\overline{\alpha}$ ($\lambda \in [0,1]$). The first subgroup of 3 objects were created with $\lambda = 0.0, 0.1, 0.2$, respectively and the second subgroup of the other 3 objects were created with $\lambda = 0.8, 0.9, 1.0$, respectively. It is easy to judge that the objects in the first subgroup are visually better than those in the second subgroup. If one subject chooses one of objects in the second subgroup within top 3 visually good objects in the user study, we regard this study invalid and discard the result from the study.

Total number of 67 subjects participated in the user study which include 28 males and 39 females. The ages of the subjects are between 16 and 65. All subjects have normal or corrected-to-normal visual acuity. Because the objects are printed in white color, we assume that the color vision capability of the subjects has no effect to our study. Three of them are invalid and we have 64 valid user study. Because these three subjects select at least one object in the second subgroup within the top 3 visually good objects of the JudgeG. The statistics of the user study is show in Figure 15. From the results, we have the following observations:

- the objects produced by our method (OurResult) are superior to DM94 and Sab96 in most cases;
- the objects produced by our method (OurResult) are comparatively as good as those produced by GroundTruth: for 3 groups (Buddha, Laurana, CleanHead), OurResult were ranked as top 1; for 4 groups (Buddha, Laurana, Lion, CleanHead), OurResult were ranked within top 2; for 5 groups (Buddha, Rabbit, Laurana, Lion, CleanHead), OurResult were ranked within top 3.

In statistics, a result is called statistically significant if it has been predicted as unlikely to have occurred by chance alone, according to a pre-determined threshold probability, the significance level α. In the following, we perform a statistical hypothesis test to draw inference using data from our user study. Table 2 lists the user study data of ranking between various methods. For individual models, each row shows the number of subjects who rank OurResult preceding the results produced by other methods. Let p be the ratio of people (not limited to subjects in user study) who consider OurResult is better than one other. For example, $p = 1/2$ means that two methods do not dominate each other. To answer such question, we state the relevant null and alternative hypotheses as follows.

$$H_0 : p = 1/2; H_1 \neq 1/2.$$

Obviously, the number of successes for OurResult in competition to the result produced by some other method, denoted by X, is a random variable which follows binomial distribution $B(n, p)$, where $n = 64$ is the total valid number of subjects in the user study.

Models	#(OR≻GroundTruth)	#(OR≻Uniform)	#(OR ≻ Random)	#(OR≻DM94)	#(OR≻Sab96)
Buddha	35	42	46	50	47
Rabbit	27	39	41	44	48
Squirrel	24	48	39	39	47
Laurana	34	45	47	54	56
Lion	28	46	41	49	46
CleanHead	35	50	52	52	46

Table 2: *The number of subjects who think OurResult is better than the results obtained by other methods. ≻ means "better than" and '#' means "the number". OR refer to our method.*

Given a significance level $\alpha = 0.05$, two numbers K_1 and K_2 can be calculated from following equations, respectively

$$\sum_{i=0}^{K_1-1} \begin{pmatrix} n \\ i \end{pmatrix} p_0^i (1-p_0)^{n-i} = \alpha/2;$$

$$\sum_{i=K_2+1}^{n} \begin{pmatrix} n \\ i \end{pmatrix} p_0^i (1-p_0)^{n-i} = \alpha/2.$$

With $p_0 = 0.5$, we get $K_1 = 25$ and $K_2 = 38$. Then we can use this region $[K_1, K_2]$ to determine what outcomes of a study would lead to a rejection of the null hypothesis for a pre-specified significance level $\alpha = 0.05$. If $K_1 \leq X \leq K_2$, we would accept the null hypothesis. If $X > K_2$ (or $X < K_1$), we would reject the null hypothesis in favor of the alternative, which means OurResult is statistically better (or worse) than the comparing one. From the user study data in Table 2, we come to the statistical inference that OurResult is comparable to GroundTruth on all models except Squirrel and dominates the results produced by the other methods (Random, Uniform, DM94 and Sab96).

Overall, we can conclude that people can hardly distinguish the objects generated by our method with corresponding ground truth objects from their appearance and our method is better than Random, Uniform, DM94 and Sab96 obviously, which means that our algorithm can save a large amount of printing time while preserving the visual salient regions of printed objects.

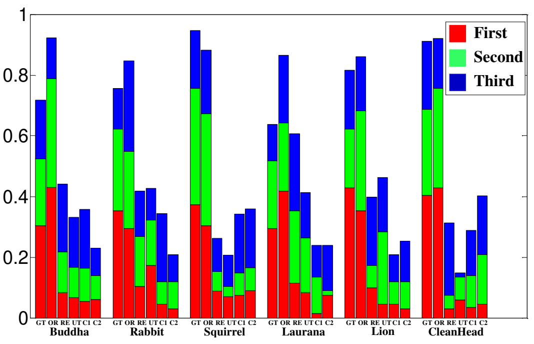

Figure 15: *The results of user study. Here the GT, OR, RE, UT, C1 and C2 are GroundTruth, OurResult, Random, Uniform, DM94 and Sab96, respectively.*

Discussions and Limitations Although we only tested our method with an FDM printer, other types of printers like Selective Laser Sintering (SLS) and Stereolithography (SLA) printers follow the same assumption of our method and can also print layers with different printable thickness in similar time. In theory, our method can be also applied to save the printing time for these printers. In practice, we didn't validate our method on SLS and SLA printers because the source code of their slicing engines and machine code is not available in public. Our segmentation algorithm saves the printing time. However, it also produces slightly visual artifacts along the vertical boundaries.

8. Conclusions and future work

In this paper, we present an efficient method for slicing 3D models for the purpose of reducing printing time while preserving the good visual salient regions as ground truth. Our adaptive layer thicknesses are generated according to the visual saliency of the 3D model which means the regions with high saliency should be sliced with high precision while the insignificant regions can be sliced with low precision. The best slicing orientation is determined based on the total printing time, volumetric error and the printing stability before slicing optimization. To further reduce the printing time, we propose a saliency based segmentation method to partition the model into subparts and each subparts is sliced individually. By managing to modify the G-code of the FDM printer, we print the object as a whole. We have tested our algorithm on a variety of 3D complex models. Both the experimental results and the user study show that our method can efficiently reduce the printing time while preserve the good visual quality of produced objects as the ground truth. For a FDM printer, the printing time is influenced by many issues in the systematic manufacture technology like acceleration and deceleration of nozzle tip during material deposition, path planning, temperature, etc. For further speed up the printing process, further studies on these issues should be conducted.

References

[AAD98] ALEXANDER P., ALLEN S., DUTTA D.: Part orientation and build cost determination in layered manufacturing. *Computer-Aided Design 30*, 5 (1998), 343–356.

[BBJP12] BÄCHER M., BICKEL B., JAMES D. L., PFISTER H.: Fabricating articulated characters from skinned meshes. *ACM*

Transactions on Graphics (Proc. SIGGRAPH) 31, 4 (2012), 47: 1–9.

[CCA*12] CALÌ J., CALIAN D. A., AMATI C., KLEINBERGER R., STEED A., KAUTZ J., WEYRICH T.: 3d-printing of non-assembly, articulated models. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2012), 130: 1–10.

[CLD*13] CHEN D., LEVIN D. I., DIDYK P., SITTHI-AMORN P., MATUSIK W.: Spec2fab: a reducer-tuner model for translating specifications to 3d prints. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 135: 1–10.

[CLM*13] CEYLAN D., LI W., MITRA N., AGRAWALA M., PAULY M.: Designing and fabricating mechanical automata from mocap sequences. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 32*, 6 (2013), 186: 1–9.

[CSPF12] CHEN X., SAPAROV A., PANG B., FUNKHOUSER T.: Schelling points on 3d surface meshes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 31*, 4 (2012), 1–11.

[CTN*13] COROS S., THOMASZEWSKI B., NORIS G., SUEDA S., FORBERG M., SUMNER R. W., MATUSIK W., BICKEL B.: Computational design of mechanical characters. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 83: 1–9.

[Cur13] CURA: Website of the source code of cura. *Available: https://github.com/daid/Cura* (2013).

[CZM*11] CHENG M.-M., ZHANG G.-X., MITRA N., HUANG X., HU S.-M.: Global contrast based salient region detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI* (2011), pp. 409–416.

[DM94] DOLENC A., MÄKELÄ I.: Slicing procedures for layered manufacturing techniques. *Computer-Aided Design 26*, 2 (1994), 119–126.

[DSDB06] DIMITROV D., SCHREVE K., DE BEER N.: Advances in three dimensional printing-state of the art and future perspectives. *Journal for New Generation Sciences 4*, 1 (2006), p–21.

[HA13] HAYASI M. T., ASIABANPOUR B.: A new adaptive slicing approach for the fully dense freeform fabrication (fdff) process. *Journal of Intelligent Manufacturing 24*, 4 (2013), 683–694.

[HBA13] HILDEBRAND K., BICKEL B., ALEXA M.: Orthogonal slicing for additive manufacturing. *Computers & Graphics 37*, 6 (2013), 669–675.

[HFW11] HAO J., FANG L., WILLIAMS R. E.: An efficient curvature-based partitioning of large-scale stl models. *Rapid Prototyping Journal 17*, 2 (2011), 116–127.

[IKN98] ITTI L., KOCH C., NIEBUR E.: A model of saliency-based visual attention for rapid scene analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1998), vol. 20, pp. 1254–1259.

[JH95] JAMIESON R., HACKER H.: Direct slicing of cad models for rapid prototyping. *Rapid Prototyping Journal 1*, 2 (1995), 4–12.

[LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: partitioning models into 3D-printable parts. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2012), 129:1–9.

[LDPT13] LAN Y., DONG Y., PELLACINI F., TONG X.: Bi-scale appearance fabrication. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (2013).

[LGB*11] LIAN Z., GODIL A., BUSTOS B., DAOUDI M., HERMANS J., KAWAMURA S., VANDERMEULEN D.: Shrec'11 track: Shape retrieval on non-rigid 3d watertight meshes. *3DOR 11* (2011), 79–88.

[LGX*13] LEVIN A., GLASNER D., XIONG Y., DURAND F., FREEMAN W., MATUSIK W., ZICKLER T.: Fabricating brdfs at high spatial resolution using wave optics. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (2013).

[LVJ05] LEE C. H., VARSHNEY A., JACOBS D. W.: Mesh saliency. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2005), vol. 24, ACM, pp. 659–666.

[Mak12] MAKERBOT: Rapid prototyping and 3D printing. `http://www.makerbot.com/`, 2012.

[MKD99] MANI K., KULKARNI P., DUTTA D.: Region-based adaptive slicing. *Computer-Aided Design 31*, 5 (1999), 317–333.

[NW06] NOCEDAL J., WRIGHT S.: *Numerical Optimization*, 2nd ed. Springer, 2006.

[PRD03] PANDEY P. M., REDDY N. V., DHANDE S. G.: Slicing procedures in layered manufacturing: a review. *Rapid Prototyping Journal 9*, 5 (2003), 274–288.

[PWLSH13] PRÉVOST R., WHITING E., LEFEBVRE S., SORKINE-HORNUNG O.: Make it stand: balancing shapes for 3d fabrication. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 81: 1–10.

[Sab96] SABOURIN E.: *Adaptive high-precision exterior, high-speed interior, layered manufacturing*. PhD thesis, Virginia Polytechnic Institute and State University, 1996.

[SLM*14] SONG R., LIU Y., MARTIN R., ROSIN P., ET AL.: Mesh saliency via spectral processing. *ACM Transactions on Graphics* (2014).

[SMKF04] SHILANE P., MIN P., KAZHDAN M., FUNKHOUSER T.: The princeton shape benchmark. *In Shape Modeling Application* (2004), 167–178.

[SVB*12] STAVA O., VANEK J., BENES B., CARR N., MĚCH R.: Stress relief: improving structural strength of 3D printable objects. *ACM Transactions on Graphics (Proc. SIGGRAPH) 31*, 4 (2012), 48:1–11.

[TB98] TYBERG J., BØHN J. H.: Local adaptive slicing. *Rapid Prototyping Journal 4*, 3 (1998), 118–127.

[VGB*14] VANEK J., GALICIA J., BENES B., MĚCH R., CARR N., STAVA O., MILLER G.: PackMerger: A 3D Print Volume Optimizer. *Computer Graphics Forum 33*, 6 (2014), 322–332.

[VWRKM13] VIDIMČE K., WANG S.-P., RAGAN-KELLEY J., MATUSIK W.: Openfab: A programmable pipeline for multi-material fabrication. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 136: 1–10.

[WSZL13] WU J., SHEN X., ZHU W., LIU L.: Mesh saliency with global rarity. *Graphical Models 75*, 5 (2013), 255–264.

[WWY*13] WANG W., WANG T. Y., YANG Z., LIU L., TONG X., TONG W., DENG J., CHEN F., LIU X.: Cost-effective printing of 3d objects with skin-frame structures. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 32*, 5 (2013), 177: 1–10.

[XWL*97] XU F., WONG Y., LOH H., FUH J., MIYAZAWA T.: Optimal orientation with variable slicing in stereolithography. *Rapid Prototyping Journal 3*, 3 (1997), 76–88.

[ZC95] ZHU P., CHIRLIAN P.: On critical point detection of digital shapes. *IEEE Transactions on Pattern Recognition and Machine Intelligence 17*, 8 (1995), 737–748.

[ZPZ13] ZHOU Q., PANETTA J., ZORIN D.: Worst-case structural analysis. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 137: 1–12.

[ZXS*12] ZHU L., XU W., SNYDER J., LIU Y., WANG G., GUO B.: Motion-guided mechanical toy modeling. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2012), 127: 1–9.