Generating sparse self-supporting wireframe models for 3D printing using mesh simplification

Liu, Xiuping; Lin, Liping; Wu, Jun; Wang, Weiming; Yin, Baocai ; Wang, Charlie

DOI
10.1016/j.gmod.2018.05.001

Publication date
2018

Document Version
Accepted author manuscript

Published in
Graphical Models

Generating Sparse Self-Supporting Wireframe Models for 3D Printing Using Mesh Simplification

Xiuping Liu[a], Liping Lin[a], Jun Wu[b], Weiming Wang[a,b,*], Baocai Yin[a], Charlie C.L. Wang[b]

[a] *School of Mathematical Sciences, Dalian University of Technology, China*
[b] *Department of Design Engineering, Delft University of Technology, The Netherlands*

Abstract

Wireframe models are becoming a popular option in 3D printing. Generating sparse wireframe models using classic mesh simplification methods leads to models that require a lot of support structures in the layer-upon-layer additive process. In this paper we present a mesh simplification method that takes into account the overhang angle. Specifically, we propose a metric for self-supportability. By combining this novel metric together with the classic error metrics for mesh simplification, our method generates sparse wireframe models that need much less supports. Moreover, the operations of vertex position optimization and edge flipping are used to further increase self-supportability of the wireframe models. We demonstrate the effectiveness of the proposed method on a number of 3D models.

Keywords: Self-supporting, Wireframe models, 3D printing, Mesh simplification

1. Introduction

Wireframe models consisting of struts are becoming a popular option in 3D printing. As an abstract representation of 3D models, a sparse wireframe model can be fabricated much faster than printing surface or solid models. This feature is especially favoured among designers. The design process typically takes multiple iterations, and a fast fabrication of intermediate versions enables quick design iteration. Wireframe structures also frequently appear in the fields of art, fashion, and architecture.

A few recent works in human-computer interaction and graphics have focused on the fabrication of sparse wireframe structures. For instance, Mueller et al. [1] proposed WirePrint to directly extrude the edges in 3D space. Tao et al. [2] presented a system for prototyping freeform surfaces composed of woven lines. Wu et al. [3] presented an algorithm for printing arbitrary meshes using a 5DOF wireframe printer. Huang et al. [4] investigated fabrication process planning in robot-assisted wireframe printing.

A problem concerning the input to the 3D printers – the design of sparse wireframe models that are more suitable for 3D printing – is less studied. In graphics faithful 3D models often have tens of thousands of edges in the surface mesh. Several established, general-purpose mesh simplification techniques (e.g., vertex clustering [5] or incremental decimation [6]) can be applied

*Corresponding author
Email address: wwmdlut@gmail.com (Weiming Wang)

to create sparse mesh structures. However, mesh simplification considering manufacturability of the simplified mesh is yet to be explored.

In this paper we present an algorithm for integrating manufacturability into mesh simplification. In particular we concern the self-supporting property in 3D printing [7]. When the angle formed by the strut and the printing direction is larger than a machine specific self-supporting angle (e.g., 45 degrees in some printers), additional structures are needed to support the strut from its beneath. Figure 1 illustrates a simple 2D wireframe model where self-supporting and non-self-supporting struts are indicated by different colours. Adding support structures prolongs the printing process and wastes material. To account for the amount of support structures, we propose a metric to measure the self-supportability. The metric is then applied to steer the mesh simplification process towards a large ratio of self-supportability. Integrated with the classic quadric error metrics (QEM) [6], our method results in sparse wireframe models that require much less supports during the additive manufacturing process.

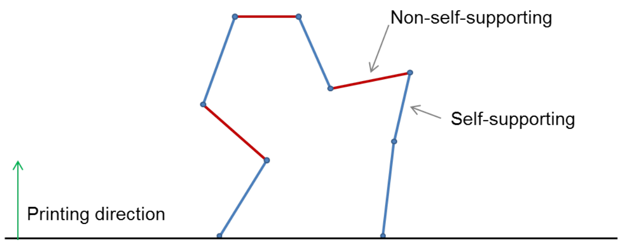


Figure 1: Illustration of a 2D wireframe model that is composed of self-supporting struts (blue) and struts that need supports (red), depending on the angle between the strut and the upright printing direction.

The contributions of this paper are summarized as follows:

- A self-supportability metric and its integration in mesh simplification to reduce supports in generating sparse wireframe models.
- A set of operations (i.e., vertex position and edge flipping) are optimized to further reduce supports.

Our method improves the self-supportability ratio, which is later defined in Eq.(1) as the ratio of self-supporting edges over total edges. An example is given in Fig. 2. The classic mesh simplification leads to a wireframe with 47.16% self-supporting edges, while our method improves the ratio to 69.79%.

2. Related Work

In this section we briefly review wireframe models in 3D printing and mesh simplification algorithms that are most closely related to our method.

3D Printing Wireframe Models Sparse wireframe models can reach a smaller volume compared to surface or solid representations of the same digital shape. The fabrication of such structures thus provides a fast preview of intermediate designs in the iterative digital content creation [1, 2]. A few recent works have focused on upgrading 3D printers dedicated to the fabrication of wireframe structures (e.g., [3, 4]). In this paper we mainly target on desktop 3D printers using *Fused Deposition Modelling* (FDM) – one of the most widely used additive technologies.

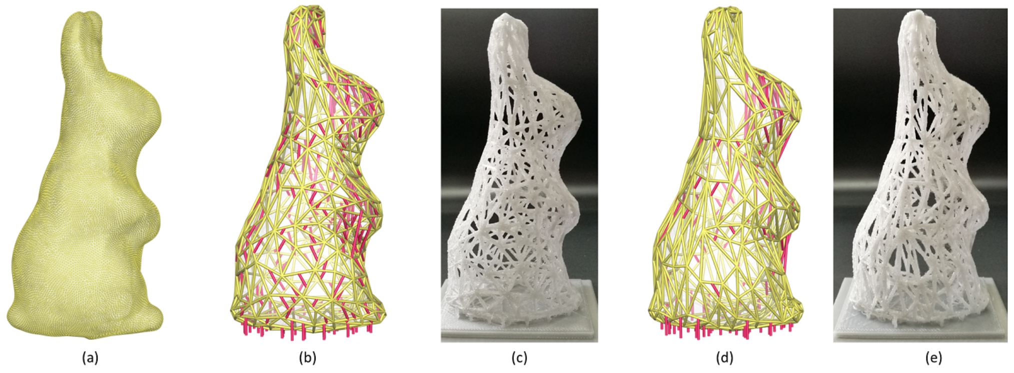


Figure 2: A dense triangle mesh with 71.8K edges (a) is simplified to 1.5K edges by the quadric error metrics [6] (b) and by the proposed method (d). The support structures added for 3D printing are indicated by red struts. The corresponding printed wireframe models are shown in (c) and (e). Our method requires much less supports. The self-supportability ratio is 69.79% by our method, and 47.16% when only using the quadric error metrics.

Wireframe structures have also been used as lightweight infill for 3D printed parts [8]. To improve the printability of mechanically optimized wireframe infill structures, Wang et al. [9] take into account the angle between struts and the printing direction. The self-supportability of optimized structures has initially been addressed in topology optimization (e.g., [7]). However, the generation of sparse wireframe models concerning both the visual appearance and the self-supportability remains as an interesting yet unsolved problem.

Mesh Simplification Digital models in graphics are often composed of tens of thousands of triangles. Directly mapping triangle edges to a wireframe model for 3D printing leads to complex structures which counteract the purpose of fast fabrication. To reduce the number of edges, we resort to mesh simplification which is a well-researched topic in graphics.

Mesh simplification methods have been maturing over the decades. Popular algorithms include vertex decimation [10, 11], vertex clustering [5, 12], subdivision meshes [13, 14], and iterative edge contraction [6, 15]. Please refer to review articles [16, 17] for detailed comparisons. Edge contraction is simple and effective, and thus it is chosen as the starting point for generating sparse self-supporting wireframe models. In particular, we adopt the quadric error metrics (QEM) proposed by Garland and Heckbert [6] for edge contraction, and enhance it with a novel metric for self-supportability.

3. Algorithm

Given a dense triangle mesh representing a 3D shape, we aim to compute a sparse triangle mesh which preserves the given 3D shape, while maximizing the ratio of self-supporting edges over all edges. Our method consists of the operations of edge contraction (Section 3.1) with an optimization of vertex position (Section 3.2) and edge flipping (Section 3.3). The resulting edges are mapped to solid cylindrical struts representing wireframe models to be fabricated.

Let us first define a few basic terms. The self-supportability of a wireframe model (M) is defined as

$$SS(M) = \frac{\sum_{e_i \in PE(M)} Len(e_i)}{\sum_{e_i \in E(M)} Len(e_i)}, \quad (1)$$

where $Len(.)$ is a function to calculate the length of an edge, $E(M)$ is the set including all edges in M, and $PE(M)$ is the subset of edges that are self-supporting. An edge (e) is classified as self-supporting if the angle ($A(e)$) formed between the printing direction and the edge falls into a specific range,

$$A(e) = \angle(\mathbf{d}(e), \mathbf{d}) \in [0, t] \cup [\pi - t, \pi], \quad (2)$$

where $\mathbf{d}(e)$ is the normalized direction of the edge and $\mathbf{d}$ is the printing direction. Without loss of generality, we assume the printing direction coincides with the upright orientation of models. The angle t is the maximally allowed angle of overhang, which depends on both the material and the manufacturing process. In our experiments, we set t to $\pi/4$.

3.1. Edge Contraction

The proposed mesh simplification algorithm is based on the iterative contraction of edges (i.e., pairs of vertices). Edge contraction is denoted by $(\mathbf{v}_1, \mathbf{v}_2) \rightarrow \bar{\mathbf{v}}$, where $\mathbf{v}_1$ and $\mathbf{v}_2$ are two endpoints of an edge (e), and $\bar{\mathbf{v}}$ is a new vertex that will replace $\mathbf{v}_1$ and $\mathbf{v}_2$. The vertices incident to $\mathbf{v}_1$ and $\mathbf{v}_2$ are then connected to the new vertex $\bar{\mathbf{v}}$ (as illustrated in Fig. 3). To decide which edges shall be contracted first, an error metric is to be defined.

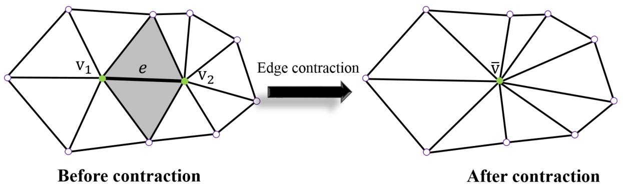


Figure 3: The edge formed by $\mathbf{v}_1$ and $\mathbf{v}_2$ is contracted and replaced by a new vertex $\bar{\mathbf{v}}$. The two shaded triangles are removed during edge contraction.

In mesh simplification we consider three factors, and each of them corresponds to a metric. The manufacturability is represented by a self-supportability metric (SSM). Visual error is measured by a normal-based metric (NBM). The shape approximation error is encoded by the classic quadric error metrics (QEM) [6]. The integrated metric for an edge, e, is thus written as

$$EM(e) = \lambda NBM(e) + \beta SSM_T(e) + QEM(e), \quad (3)$$

where the weighting factors λ and β can be set by users to balance between these three factors.

Edges are first sorted in the ascending order of error $EM(e)$. We then iteratively contract edges and update the metrics locally until a user-specified terminal condition is achieved – i.e., regarding the remaining edges.

Self-Supportability Metric (SSM). SSM measures the change in self-supporting property caused by edge contraction, $(\mathbf{v}_1, \mathbf{v}_2) \rightarrow \bar{\mathbf{v}}$. Denote the self-supporting length ratio of the new vertex by $R_{ss}(\bar{\mathbf{v}})$, and the self-supporting length ratio of the edge by $R_{ss}(e)$. The metric is

$$SSM(e) = R_{ss}(\bar{\mathbf{v}}) - R_{ss}(e). \quad (4)$$

$R_{ss}(\mathbf{v})$ defines, for a vertex $\mathbf{v}$, the self-supporting length ratio of its incident edges as

$$R_{ss}(\mathbf{v}) = \frac{\sum_{e_i \in Pe(\mathbf{v})} Len(e_i)}{\sum_{e_i \in Ne(\mathbf{v})} Len(e_i)}. \tag{5}$$

Here, $Ne(\mathbf{v})$ is the set of edges incident to vertex $\mathbf{v}$ and $Pe(\mathbf{v})$ is the subset of $Ne(\mathbf{v})$ that consists of edges that are printable (i.e., self-supporting).

For an edge e, its self-supporting length ratio is defined from its two vertices ($\mathbf{v}_1$ and $\mathbf{v}_2$) as

$$R_{ss}(e) = R_{ss}(\mathbf{v}_1 \cup \mathbf{v}_2) = \frac{\sum_{e_i \in Pe(\mathbf{v}_1) \cup Pe(\mathbf{v}_2)} Len(e_i)}{\sum_{e_i \in Ne(\mathbf{v}_1) \cup Ne(\mathbf{v}_2)} Len(e_i)}, \tag{6}$$

A larger value of $SSM(e)$ means that contracting e to $\bar{\mathbf{v}}$ will significantly increase the self-supportability, and thus shall be favoured. To make a consistent priority list with error metrics QEM and NBM to be introduced later, we transform it by an exponential function as

$$SSM_T(e) = e^{-SSM(e)}. \tag{7}$$

With this transformation, the operation of edge contraction starts with the lowest error values.

Normal-Based Metric (NBM). The field of normal vectors is fundamental in the way the 3D shape is perceived. Normal vectors govern lighting effects such as diffusion, specularity, as well as curvature lines and silhouettes, and discontinuity of normal indicates geometric features. To account for this in mesh approximation, Cohen-Steiner et al [18] proposed a $\mathcal{L}^{2,1}$ shape metric based on normal to cluster faces with similar normal vectors. Following this idea, we use the normal discrepancy before and after contraction to measure the visual error caused by edge contraction – that is

$$NBM(e) = \frac{\sum_{f \in Nf(\bar{\mathbf{v}})} \|\mathbf{n}_f - \mathbf{n'}_f\|}{\#(Nf(\bar{\mathbf{v}}))}, \tag{8}$$

where $Nf(\bar{\mathbf{v}})$ is the set of triangle faces associated to the new vertex $\bar{\mathbf{v}}$, #(.) is the cardinal of a set, and $\mathbf{n}_f$ and $\mathbf{n'}_f$ are normal vectors of the corresponding faces before and after edge contraction respectively.

The normal-based metric Eq.(8) measures the change in normal caused by the edge contraction of e. The edge with the smallest $NBM(e)$ value has the highest priority to be collapsed.

Quadric Error Metrics (QEM). We use the quadric error metrics proposed by Garland and Hechbert [6] to measure the geometric error caused by edge contraction. For completeness we briefly introduce the QEM. The error of the vertex $\mathbf{v}$ is defined as the sum of squared distances to its associated planes (triangles) as

$$\triangle(\mathbf{v}) = \mathbf{v}\mathbf{Q}\mathbf{v}^{\mathbf{T}} = \sum_{\mathbf{p} \in planes(\mathbf{v})} (\mathbf{p}^{\mathbf{T}}\mathbf{v})^2, \tag{9}$$

where $planes(\mathbf{v})$ is the set of planes of the triangles that meet at vertex $\mathbf{v}$, and $\mathbf{p} = [a\ b\ c\ d]^T$ represents the plane defined by the equation $ax + by + cz + d = 0$ with $a^2 + b^2 + c^2 = 1$. Notice that Eq.(9) can be rewritten as a quadric form:

$$\triangle(\mathbf{v}) = \mathbf{v}^T(\sum_{\mathbf{p} \in planes(\mathbf{v})} \mathbf{K}_{\mathbf{p}})\mathbf{v}, \tag{10}$$

where $\mathbf{K_p}$ is the matrix:

$$\mathbf{K_p} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}. \tag{11}$$

$\mathbf{K_p}$ is assembled for representing the entire set of planes by a single matrix $\mathbf{Q}$. Then, the geometric error caused by edge contraction is defined as:

$$QEM(e) = \bar{\mathbf{v}}^T(\mathbf{Q}_1 + \mathbf{Q}_2)\bar{\mathbf{v}}, \tag{12}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are quadric matrices of $\mathbf{v}_1$ and $\mathbf{v}_2$ respectively. The edge with the smallest $QEM(e)$ has the highest priority to be simplified.

3.2. Vertex Position Optimization

An important component in edge contraction is to find the position of the newly created vertex $\bar{\mathbf{v}}$. Simple options for the new vertex include the endpoints or the middle point of the edge. The position can also be optimized to reduce geometric errors by solving a linear function. In this work, we optimize the position for an increased self-supportability.

Our optimization aims to find, between the two endpoints $\mathbf{v}_1$ and $\mathbf{v}_2$ of edge e, a vertex $\bar{\mathbf{v}}$ which has the largest $SSM(e)$. Let us express the position by $\bar{\mathbf{v}} = \mathbf{v}_1 + x(\mathbf{v}_2 - \mathbf{v}_1)$, $x \in [0, 1]$. Considering that the term $R_{ss}(e)$ in $SSM(e)$ is constant, the optimization problem is formulated as

$$\max_x \quad R_{ss}(\mathbf{v}_1 + x(\mathbf{v}_2 - \mathbf{v}_1)), \tag{13}$$

$$\text{s.t.} \quad x \in [0, 1]. \tag{14}$$

The optimization is solved by using the interior point method.

3.3. Edge Flipping

After mesh simplification using edge contraction, edges are evaluated for flipping to further improve the self-supportability. Edge flipping is found beneficial in the following two cases. First, in Fig. 4 (top left) the red edge is impending and needs extra support structures during printing. However, if it is flipped to the other direction, it becomes self-supporting (the blue edge in Fig.4 (top right)). A second case is as illustrated in Fig. 4 (bottom). Even if the flipping does not directly lead to a self-supporting edge, it reduces the angle between the edge and the upright printing direction and therefore can be beneficial for the following flipping operations during the recursive flipping process.

We note that edge flipping may create an intersection with other edges or increase the geometric error. For the situation illustrated in Fig. 5 (top) where all edges are almost coplanar, flipping the red edge into the blue one creates an intersection with the other edges. In another example shown in Fig. 5 (bottom), the red edge indicates a sharp feature in the original mesh. Flipping it into the blue one on the right leads to a large geometric error. Therefore, geometric error and local shape are considered as well when evaluating the edges for flipping.

To this end, the conditions for flipping the edge e are summarized as follows.

- The dihedral angle between the two faces sharing e is larger than a threshold γ ($\gamma = 160$ degree is used in our experiments).

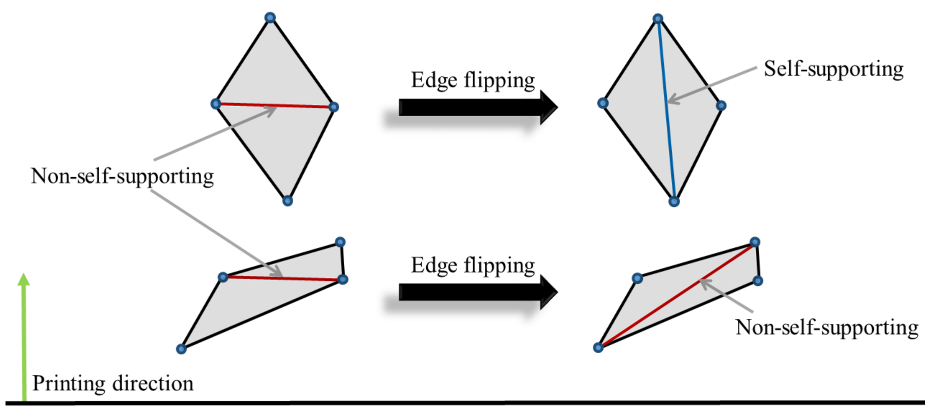


Figure 4: Top: The red edge on the left is impending. Flipping it creates a self-supporting edge on the right. Bottom: Flipping does not create a self-supporting edge, it reduces the angle between the edge and the upright printing direction.

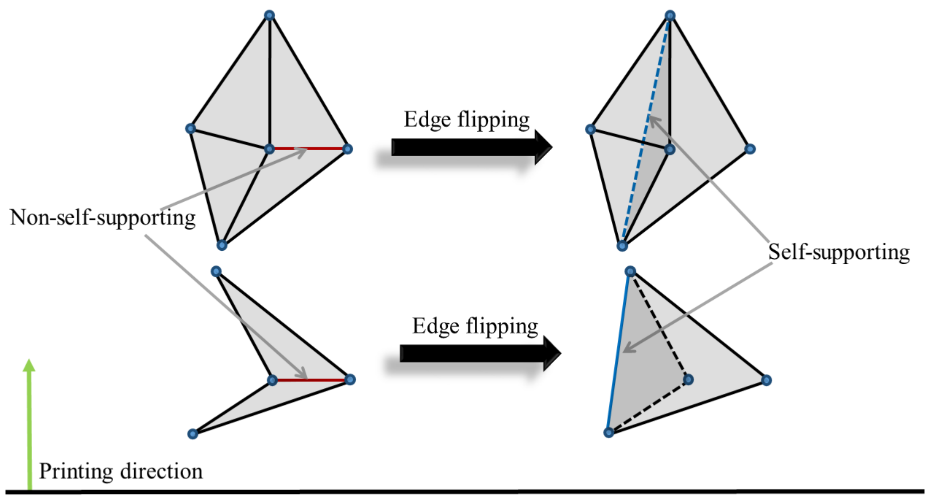


Figure 5: While flipping the red edges on the left to the blue edges improves the printability, it creates edge intersection, or a large geometric error.

- Projecting the two adjacent faces onto a plane, the obtained polygon is convex.
- Flipping increases the absolute value of the inner product between the normalized edge direction and the printing direction, which means an increase in self-supportability.

4. Results

We have implemented the proposed method using C++, while calling Matlab functions to solve the optimization of vertex position (Eq. 13). It has been tested on a desktop PC with Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz and 32GB memory.

4.1. Parameter Analysis

We study the influence of the threshold angle γ and the target number of edges, and analyze different simplification options and metrics. For all our tests the weighting factors in the metric Eq. 3 are $\lambda = 0.7$ and $\beta = 0.2$.

Threshold angle. The threshold angle γ in edge flipping has a significant influence on the self-supportability of the model, $SS(M)$. An example is given in Fig. 6 on the Rabbit model. The simplified model has 1.5K edges. From left to right, the γ value is 80^o, 100^o, 120^o, 140^o, and 160^o, respectively. The corresponding $SS(M)$ values are: 87.74%, 84.22%, 81.03%, 76.04%, and 69.79%. It can be observed that a smaller γ value (e.g., 80^o) leads to a larger $SS(M)$ value. Yet, the corresponding meshes may become distorted. In contrast, a large γ value better preserves the shape at the cost of a lower $SS(M)$ value. While this parameter can be tuned by the user, it is set as $\gamma = 160^o$ in our following tests.

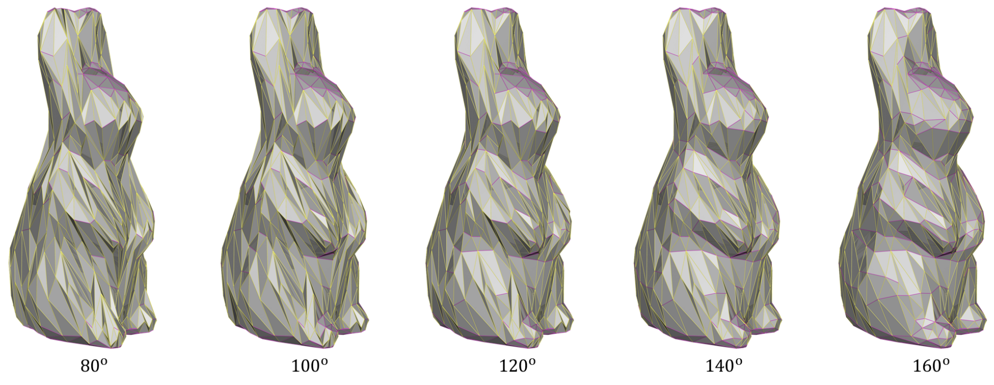


Figure 6: Edge flipping is applied on a Rabbit model. The flipping threshold angle γ increases from left to right. The self-supportability ratios $SS(M)$ from left to right are: 87.74%, 84.22%, 81.03%, 76.04%, and 69.79% respectively.

Number of simplified edges. Fig. 7 shows simplified Jar models with different numbers of edges. The original model has 66.7K edges, and it is simplified to, from left to right, 200, 500, 1000, 1500, and 2000 edges respectively. The $SS(M)$ values vary slightly, in the range between 61.79% and 63.86%. To measure the geometric error of the simplified model, we compute the Hausdorff distance between the simplified mesh and the original mesh. The error decreases as the number of edges increases: 13.20, 9.33, 7.82, 7.18, and 6.92 respectively. In this paper, most models are simplified to 1500 edges.

Simplification options. Our algorithm consists of three parts which can be combined: Error metric (EM), vertex position optimization (VPO), and edge flipping (EF). Figure 8 shows simplified meshes from the Armadillo model. The original model (a) has 95.9K edges, and it is simplified to 1.5K edges in (b-e). From (b) to (e), it shows the simplified meshes generated by using four different strategies: (b) QEM [6]; (c) EM ; (d) EM + VPO; (e) EM + VPO + EF. The struts are rendered in two different colours, with yellow indicating self-supporting struts and purple for non-self-supporting struts. The $SS(M)$ values are 39.22%, 45.91%, 50.27%, 59.87%, and 65.72% respectively. This comparison confirms that the combination of EM, VPO and EF achieves the highest self-supportability.

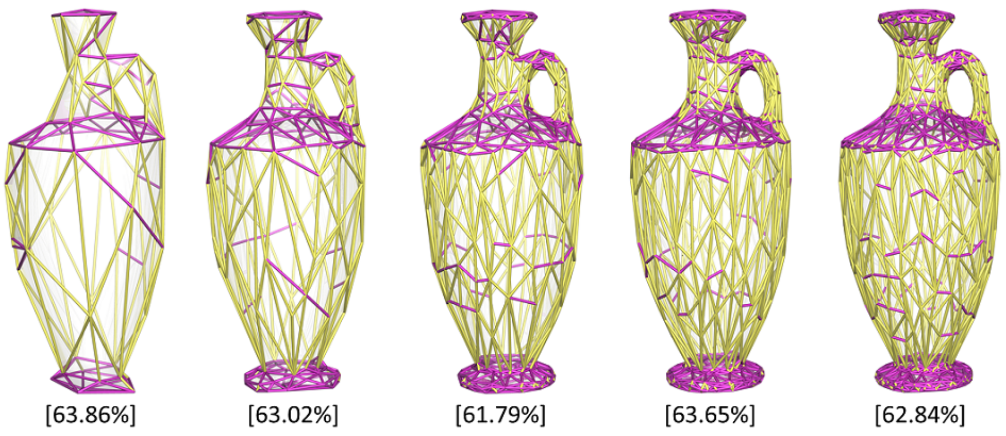


Figure 7: A Jar model is simplified to, from left to right, 200, 500, 1000, 1500, and 2000 edges, respectively. The $SS(M)$ value for each model is reported.

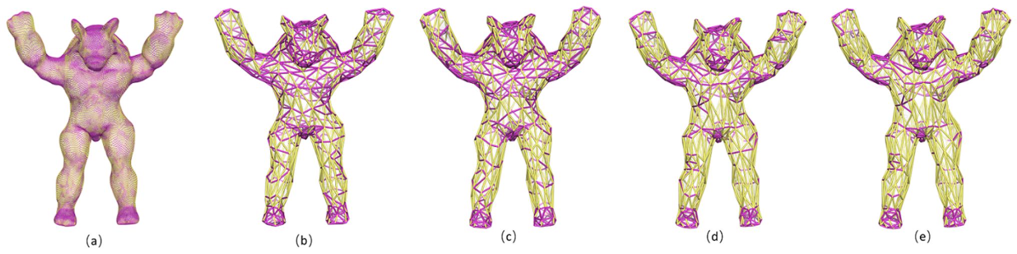
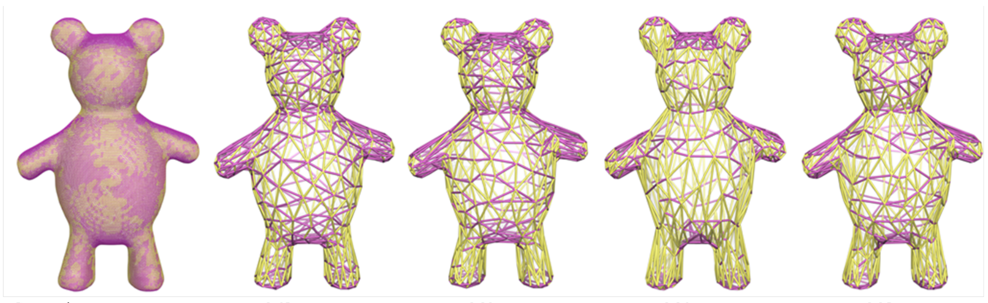


Figure 8: (a) is the input mesh. (b) is obtained by QEM [6]. (c)-(e) are generated by our algorithm with different strategies: (c) with the proposed error metric (EM), (d) with EM and vertex position optimization (VPO), and (e) with EM, VPO and edge flipping. The self-supportability values $SS(M)$ are: (a) 39.22%, (b) 42.34%, (c) 43.98%, (d) 51.18%, and (e) 60.40% respectively.

Integrated metric. Our integrated metric (EM) builds upon the quadric error metrics (QEM) and additionally considers a normal-based error metric (NBM), and a self-supportability metric (SSM). Figure 9 shows the results from different combinations of these sub-metrics. Edge flipping and vertex optimization are not activated in this comparison. We compare the simplified models regarding the self-supportability ratio and the geometric error, which is defined by the Hausdorff distance between the input and simplified meshes. Figure 9 (c) has the smallest geometric error, demonstrating the effectiveness of the normal-based error metric for measuring the geometric error. With the self-supportability metric integrated, Fig. 9 (d) achieves the highest self-supportability ratio. The integration of NBM and SSM (as shown in Fig. 9 (e)) achieves a good self-supportability ratio close to Fig. 9 (d), and meanwhile a small geometric error close to Fig. 9 (c) and outperforms Fig. 9 (b) regarding both indicators.



Geometric error:	3.65	3.29	3.86	3.35
Self-supportability ratio :	45.47%	51.12%	63.29%	58.89%
(a) Input model	(b) QEM	(c) QEM+NBM	(d) QEM+SSM	(e) QEM+NBM+SSM

Figure 9: The input Bear model is simplified with different combinations of sub-metrics. The combination (e) achieves a good trade-off between geometric error and self-supportability: It has a good self-supportability ratio close to (d) and meanwhile a small geometric error close to (c).

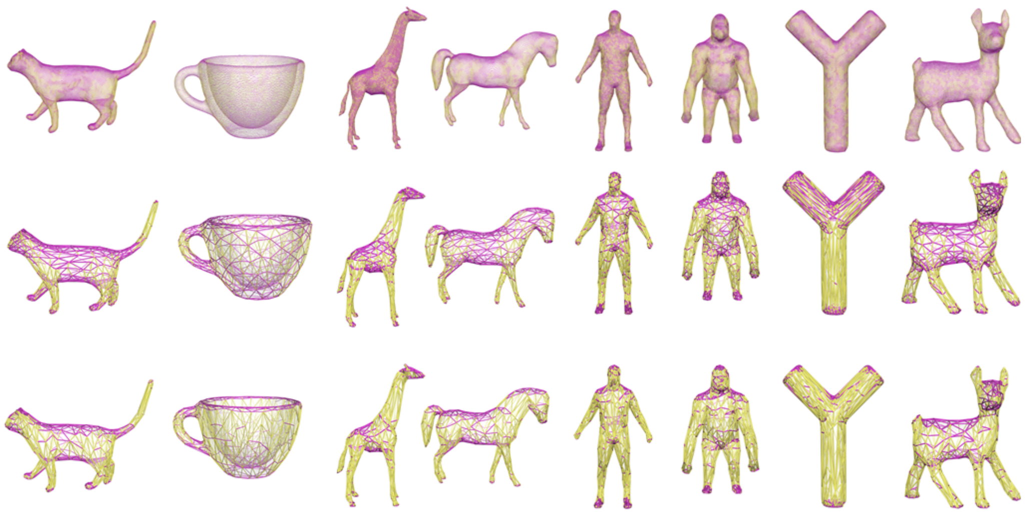


Figure 10: A set of input models (top) is simplified by edge contraction using QEM (middle) and by the proposed method (bottom). Self-supporting struts are rendered by the yellow edges, while non-self-supporting ones by the purple edges.

4.2. Comparison

We report results in comparison to QEM [6], since the proposed method extends this classic method. Fig. 10 shows, from top to bottom, the input mesh, the mesh simplified by QEM, and the mesh simplified by the proposed algorithm. The yellow frames represent self-supporting struts, while the purple frames are non-self-supporting struts. It can be clearly seen that the proposed algorithm results in a smaller number of non-self-supporting struts. An open surface mesh is tested in Fig. 11, which also confirms the effectiveness of our method.

Figure 12 shows the cumulative distribution of edges for the absolute value of the inner

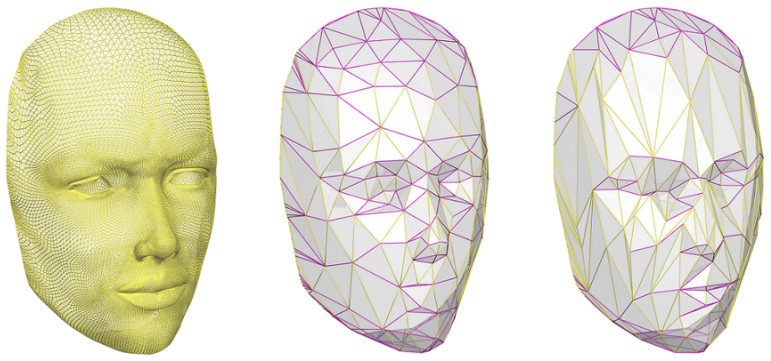


Figure 11: From left to right: The input face model, the models simplified by QEM [6] and by the proposed algorithm. Both simplified models have the same number of edges: 400. The SS(M) values are: 43.95% for QEM and 67.64% for our method.

product between the normalized edge direction and printing direction. The blue and red curves correspond to models simplified by QEM and our algorithm respectively. A peak in the red curve appears at 0.707 which corresponds to the overhang threshold $\pi/4$ set in our tests. On the left of the peak, the red curves are below the blue curves, suggesting a smaller number of non-self-supporting edges in the models simplified by our algorithm. On the right of the peak, the red curves are above the blue curves, i.e., a larger number of self-supporting edges are obtained by our algorithm.

We further evaluate the proposed normal-based metric in edge contraction. The normal-based metric considers the change in the normal vectors of faces. This can preserve thin features in the models. In Fig. 13, a Camel model with 59.9K edges is simplified to a model with 1.5K edges by both QEM and our method. It can be seen that some thin features (the tail and horns) disappear in QEM simplified mesh, while they are well preserved by our method. In this test, the $SS(M)$ values are: 57.86% for QEM and 70.18% for our method.

Table 1 details the computational statistics on the tested models. The heights of the tested models are scaled to 80 mm. The second column compares the number of edges before and after mesh simplification. The number of initial edges is between 47K and 115K, and it is simplified to 1.5K. Following that, the group composed of five columns reports the $SS(M)$ value: from left to right, the original mesh, QEM, and three variants of our method (i.e., EM, EM+VPO, and EM+VPO+EF). The statistics show that our method achieves an improvement of about 20% in the ratio compared to QEM. The eighth and ninth columns compare QEM and our method on the geometric error and the total length of edges respectively. The geometric error is represented by the Hausdorff distance between the simplified mesh and the original mesh. The data suggests that our method has a slightly larger error, and has a total length comparable than QEM. The last column reports the computation time. Our method is fast, and most models can be simplified within one minute.

4.3. Discussion

Our method achieves a large self-supportability ratio. The threshold angle γ in edge flipping can be reduced to get an even larger ratio by creating more inclined edges. However, it comes at the cost of mesh distortion. See the inset on the right side for an example. The setting of $\gamma = 0$

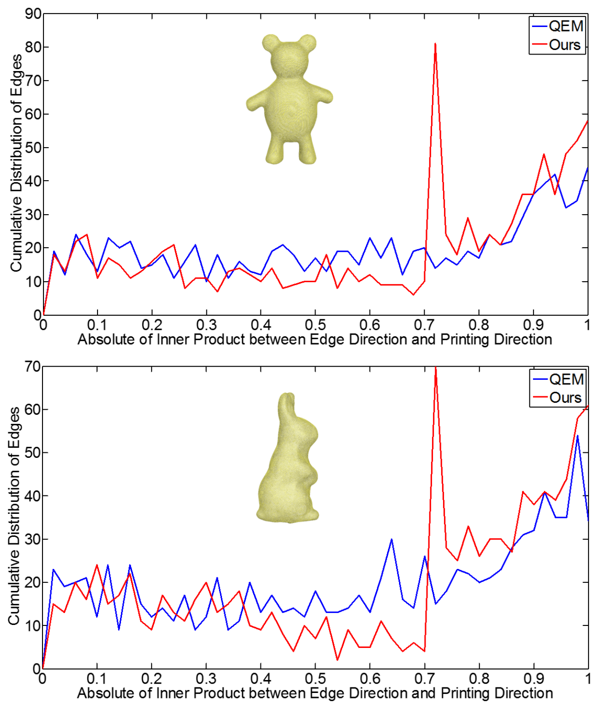


Figure 12: The cumulative distribution of edges for the absolute of the inner product between the normalized edge direction and the printing direction. The blue and red curves correspond to models simplified by QEM and our algorithm respectively.

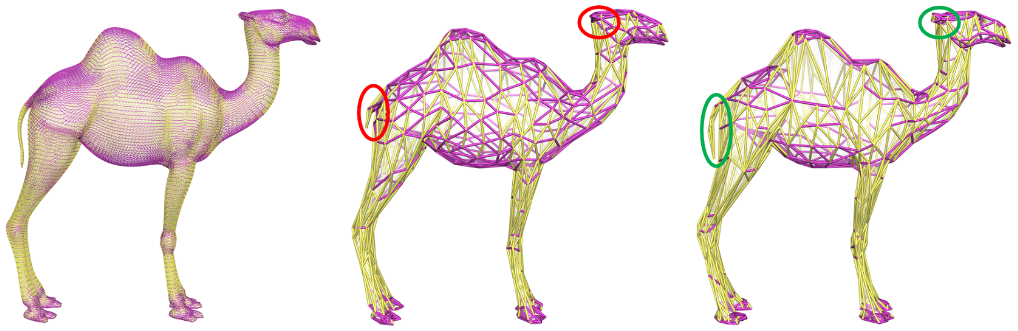


Figure 13: From left to right: The input Camel model (with 59.9K edges), the models simplified by QEM [6] and by our method. Both simplified models have the same number of edges (i.e., 1.5K). Our method is able to preserve thin geometric features, highlighted by the circles. The yellow struts are self-supporting, while the purple struts are non-self-supporting.

achieves a $SS(M)$ value of 95%, but leading to many sliver triangle faces. From a practical perspective, the trade-off between visual quality and printing speed can be assessed by the designer, as the mesh simplification usually takes less than a minute.

Table 1: Computational statistics of our approach. ”Ori” and ”Sim” represent the original mesh and the simplified mesh, respectively. ”Self-supportability Ratio” is calculated as the length of self-supporting edges over the total length of the mesh. ”Geo Error” is the Hausdorff Distance between the simplified meshes and the original meshes. ”Len” represents the total edge length of the model simplified with QEM [6] and our method.

Model	#Edge(Ori.)/	Self-supportability Ratio (%)					Geo. Error (mm)	Len. ($\times 10^3$ mm)	Timing
	#Edge(Sim.)	Ori.	QEM	EM	EM+VPO	EM+VPO+EF	QEM/Ours	QEM/Ours	(sec.)
Y_model	47.9K/1.5K	39.51	58.76	67.12	71.99	82.35	6.68/8.87	7.08/7.40	32.7
Armodino	95.9K/1.5K	39.22	45.91	50.27	59.87	65.72	5.21/6.41	6.70/7.23	50.2
Bunny	114.5K/1.5K	39.80	45.47	52.09	58.89	67.73	4.34/5.35	6.22/6.46	71.9
Camel	59.9K/1.5K	53.30	57.86	59.99	66.70	70.18	5.89/8.71	7.25/7.68	36.5
Cat	95.9K/1.5K	38.20	41.46	47.43	56.53	63.34	7.35/6.44	7.30/7.83	57.8
Cup	45.1K/1.5K	32.79	40.86	41.06	51.84	65.14	10.45/12.30	17.02/18.85	28.1
Giraffe	110.8K/1.5K	38.81	67.63	71.19	77.84	80.04	3.88/5.26	5.05/5.36	68.9
Horse	59.9K/1.5K	42.91	51.05	55.56	63.66	70.16	6.50/9.08	8.25/8.81	36.4
Human	60.5K/1.5K	43.97	66.36	71.02	76.74	80.82	4.15/5.43	5.20/5.59	36.5
Pongo	71.8K/1.5K	52.20	61.11	65.07	73.07	79.05	4.46/5.20	6.53/7.26	45.2
Woman	63.5K/1.5K	45.32	51.45	59.03	65.31	71.53	3.97/5.31	5.21/5.71	38.6
Sheep	46.9K/1.5K	37.54	52.87	57.88	62.72	70.22	5.53/5.47	6.84/7.35	29.3
Rabbit	71.8K/1.5K	41.96	47.16	52.10	59.75	69.79	5.52/6.16	6.37/6.99	45.5
Jar	66.7K/1.5K	36.13	49.36	50.43	57.18	63.65	5.47/7.17	6.26/6.70	13.2

We use a simple strategy to generate supports for the non-self-supporting struts. For the lower node of non-self-supporting strut, we search for a supporting node in its neighbourhood, such that the edge connecting the two nodes is self-supporting. If no supporting node can be identified, a vertical edge will be created to support the impending node. The effectiveness of the mesh simplification and supports generation is confirmed by 3D printing using the FDM technology. Fig. 2 shows two prints of the Rabbit model. Furthermore, Fig. 14 shows prints of another four wireframe models simplified by our algorithm.

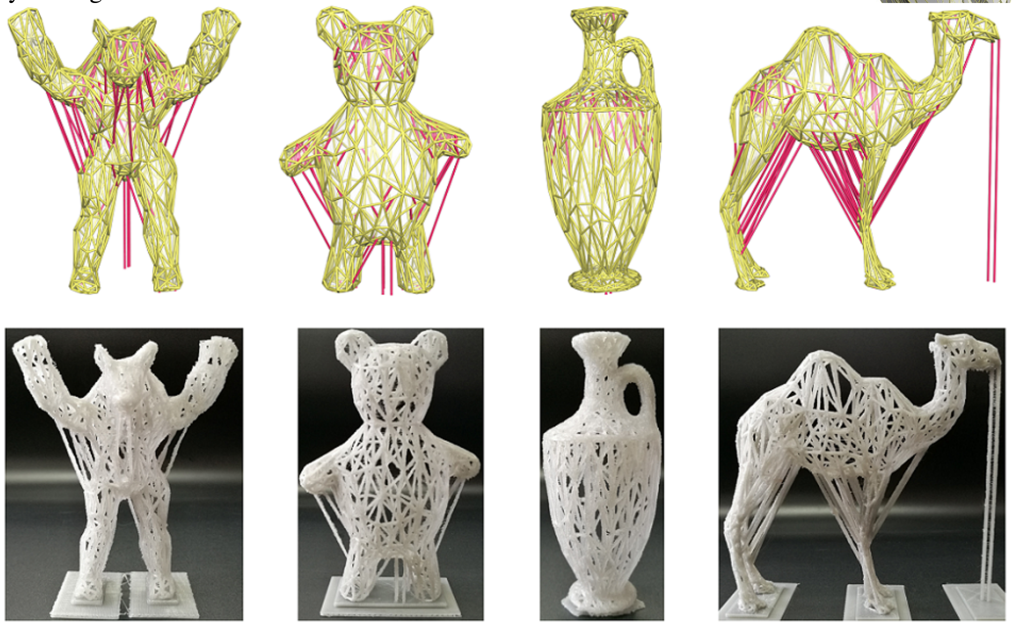


Figure 14: In the first row, the additional supports are added for the wireframe models generated by our method. The corresponding printed objects are shown in the second row.

5. Conclusion

We have presented an edge contraction-based mesh simplification algorithm which considers geometric errors, changes of normal vectors, and self-supportability of edges. The position of the contracted edge is optimized for a higher self-supportability by our method. Furthermore, edge flipping is also applied to increase the self-supportability. On a number of tested 3D models, the proposed algorithm can achieve an improvement of 14% to 25% on the self-supportability, compared to the classic method based on the quadric error metrics [6].

In this paper we reduce the number of frames by mesh simplification. Re-meshing (e.g., [19]) or mesh approximation (e.g., [18]) is an alternative strategy which potentially improves mesh quality. It is unclear yet whether a self-supportability metric can be efficiently integrated into these approaches. We leave this as future work. Another direction we would like to pursue is to extend from the self-supportability ratio to more comprehensive measures of the fabrication cost, e.g., the volume of support structures and the printing speed.

Acknowledgment

Weiming Wang would thank the support of the Natural Science Foundation of China (61702079), China Postdoctoral Science Foundation (2016M601308), and Fundamental Research Fund (DUT16 RC(3)061). This work is also supported by the Natural Science Foundation of China (61370143, 61432003, 61562062). C.C.L. Wang would also thank the support of Hong Kong RGC GRF Grant (14207414) and Natural Science Foundation of China (NSFC) (Ref. No.: 61628211).

References

[1] S. Mueller, S. Im, S. Gurevich, A. Teibrich, L. Pfisterer, F. Guimbretière, P. Baudisch, Wireprint: 3d printed previews for fast prototyping, in: Proceedings of the 27th annual ACM symposium on User interface software and technology, ACM, 2014, pp. 273–280. doi:10.1145/2642918.2647359.

[2] Y. Tao, G. Wang, C. Zhang, N. Lu, X. Zhang, C. Yao, F. Ying, Weavemesh: A low-fidelity and low-cost prototyping approach for 3d models created by flexible assembly, in: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI '17, ACM, New York, NY, USA, 2017, pp. 509–518. doi:10.1145/3025453.3025699.

[3] R. Wu, H. Peng, F. Guimbretière, S. Marschner, Printing arbitrary meshes with a 5dof wireframe printer, ACM Transactions on Graphics (TOG) 35 (4) (2016) 101. doi:10.1145/2897824.2925966.

[4] Y. Huang, J. Zhang, X. Hu, G. Song, Z. Liu, L. Yu, L. Liu, Framefab: Robotic fabrication of frame shapes, ACM Transactions on Graphics (TOG) 35 (6) (2016) 224. doi:10.1145/2980179.2982401.

[5] K.-L. Low, T.-S. Tan, Model simplification using vertex-clustering, in: Proceedings of the 1997 symposium on Interactive 3D graphics, ACM, 1997, pp. 75–ff. doi:10.1145/253284.253310.

[6] M. Garland, P. S. Heckbert, Surface simplification using quadric error metrics, in: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1997, pp. 209–216. doi:10.1145/258734.258849.

[7] J. Wu, C. C. Wang, X. Zhang, R. Westermann, Self-supporting rhombic infill structures for additive manufacturing, Computer-Aided Design 80 (2016) 32 – 42. doi:10.1016/j.cad.2016.07.006.

[8] W. Wang, T. Y. Wang, Z. Yang, L. Liu, X. Tong, W. Tong, J. Deng, F. Chen, X. Liu, Cost-effective printing of 3d objects with skin-frame structures, ACM Transactions on Graphics (TOG) 32 (6) (2013) 177. doi:10.1145/2508363.2508382.

[9] W. Wang, S. Qian, L. Lin, B. Li, B. Yin, L. Liu, X. Liu, Support-free frame structures, Computers & Graphics 66 (2017) 154–161. doi:10.1016/j.cag.2017.05.022.

[10] W. J. Schroeder, J. A. Zarge, W. E. Lorensen, Decimation of triangle meshes, in: ACM Siggraph Computer Graphics, Vol. 26, ACM, 1992, pp. 65–70. doi:10.1145/142920.134010.

[11] K. J. Renze, J. H. Oliver, Generalized unstructured decimation [computer graphics], IEEE Computer Graphics and Applications 16 (6) (1996) 24–32. doi:10.1109/38.544069.

[12] J. Rossignac, P. Borrel, Multi-resolution 3d approximations for rendering complex scenes, Modeling in computer graphics: methods and applications 465. `doi:10.1007/978-3-642-78114-8_29`.

[13] I. Guskov, K. Vidimče, W. Sweldens, P. Schröder, Normal meshes, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 95–102. `doi:10.1145/344779.344831`.

[14] M. Lounsbery, T. D. DeRose, J. Warren, Multiresolution analysis for surfaces of arbitrary topological type, ACM Transactions on Graphics (TOG) 16 (1) (1997) 34–73. `doi:10.1145/237748.237750`.

[15] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Mesh optimization, in: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, ACM, 1993, pp. 19–26. `doi:10.1145/166117.166119`.

[16] P. Cignoni, C. Montani, R. Scopigno, A comparison of mesh simplification algorithms, Computers & Graphics 22 (1) (1998) 37–54. `doi:10.1016/S0097-8493(97)00082-4`.

[17] D. P. Luebke, A developer's survey of polygonal simplification algorithms, IEEE Computer Graphics and Applications 21 (3) (2001) 24–35. `doi:10.1109/38.920624`.

[18] D. Cohen-Steiner, P. Alliez, M. Desbrun, Variational shape approximation, ACM Trans. Graph. 23 (3) (2004) 905–914. `doi:10.1145/1015706.1015817`.

[19] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, M. Desbrun, Anisotropic polygonal remeshing, ACM Trans. Graph. 22 (3) (2003) 485–493. `doi:10.1145/882262.882296`.