Robust Voxelization of Arbitrary Meshes

[1,*]Hung-Kuang Chen and [1]Chi-Yuan Tu

Abstract

Fast sliced-based voxelization has attracted much attention in recently years for its excellence in runtime efficiency. However, most of the proposed methods failed to derive correct results from non-manifold meshes containing boundaries or interpenetrations. Utilization of such incorrect results may fail for a number of further applications such as thinning and collision detections, etc. To address this issue, we begin with a process called vertex grouping to find maximum connected components from the input mesh, followed by applying a slice-based solid voxelization to each part of the input mesh. The final voxel set is then derived by merging voxels of all the parts. According to the experimental results, the new method is capable of deriving robust continuous volume data with no erroneous and redundant voxels, which is useful for a number of further applications such as the collision detection and skeleton extraction.

Keywords: planetary gear train, cam, innovation design, variable speed mechanism.

*Corresponding Author: Hung-Kuang Chen
(E-mail: hank@ncut.edu.tw)
[1] Electronic Engineering Department, National Chin-Yi University of Technology, Taichung, Taiwan 40109.

1. Introduction

Recently the innovation of fast graphics accelerators and efficient volume rendering techniques introduced an explosive growth of volume rendering related applications. Consequently, real-time voxelization of the surface meshes as one of the essential techniques of volume rendering has gained much attention and was intensively studied.

The concept of scanning convert 3D geometric objects into their discrete voxel representation was first introduced by Arie Kaufman in 1986[13]. In comparison with the surface based representations, volumetric representation captures not only the surface details but also the internal variations of a 3D object. According to an overview done by Arie Kaufman in [12], for common volume rendering approaches, the input volume data are usually presented in a discrete form, called voxels, as its direct or intermediate representation, which could be collected either by computing from scientific visualization or by sampling the real-world objects using sensing technologies such as CT, MRI, remote sensing, 3D range scanning, etc.

Despite the traditional applications in medial image visualization domain [9], the focus of voxel-based representation applications has been on the improvement of the runtime efficiency of collision detection [8][1] and the curve skeleton extraction [16][21]. However, most current

frame-based techniques accept only manifold meshes. For non-manifold meshes or those consisting of intersecting parts such as the Al Capone model as shown in Fig. 1, the computed result is very likely to be incorrect.

Figure 1: The Al Capone mesh.

According to our study, there exists no feasible solution to effectively deal with such problems so far. To cope with these problems, we proposed a novel voxelization method adaptable to arbitrary meshes, called adaptive voxelization. The new method begins with a process called vertex grouping to segment the input mesh into disjoined parts followed by applying a render-based solid voxelization to each segmented parts of the input mesh. The final result, namely, the complete voxel set, is then derived by merging all the voxels.

To address these issues, we propose a novel robust approach to surface mesh voxelization. The new method is able to voxelize both manifold and non-manifold meshes as well as those consisted of intersecting parts. Moreover, with our new method, there is no need to repair cracked meshes in advance to ensure that the computed results contains no excess incorrect voxel outside the model range.

2. Terminologies

Prior to the discussion of our approach, we briefly introduce a number of related notations and terminologies as follows[15].

Assuming an object is initially be represented by a polygonal mesh M = (V, F), where V is a set of points represented by a 3-tuple of real numbers, (v_x, v_y, v_z), in 3D Euclidean space R^3; F is a set of triangular faces F defined by a subset of three different vertices of V represented as a 3-tuple of vertex indices, f = (I_a, I_b, I_c) $\in I^3$, to the vertices of V.

Definition 1 The **voxelization** of a mesh M is concerned with the discretization of M into a set of voxels S that ”best” represents M within the discrete voxel space $\Sigma \subset Z^3$.

Definition 2 A **voxel** is a unit cubic volume centered at a grid point of a 3D discrete space Z3, which is assigned explicitly with values of either ‘1’ or ‘0’.

Definition 3 A voxel of value ‘1’ is called a **black** or a **foreground voxel** indicating the presence an opaque sample.

Definition 4 A voxel with ‘0’ value is called a **white** or a **background voxel** corresponding to a void space or the presence of a transparent sample.

In particular, we denote the universal set of voxels, the set of black voxels, and the set of white voxels respectively as , Σ, S, and Σ − S.

3. Related Works

The methods of voxelization can be classified into two types, i.e., the surface voxelization and the solid voxelization. In surface voxelization; only the boundary voxels corresponding to the surfaces are recorded, while in solid voxelization the interior voxels corresponding to the inner parts of the model are also kept in the final result.

In [11], Aggeliki Karabassi et. al. proposed a way to do voxelization by making use of the depth values stored in the Z-buffer, which measures the maximum and minimum distance to each face of the bounding box by parallel projections. However, this method may result in incorrect voxels if the object has hidden cavities.

An improvement toward better computational efficiency using hardware acceleration was reported later by Fang and Chen [5][6]. Their method started with slicing the volume space defined by the bounding box covering the object along the directions of the three coordinate axes, followed by rendering the sliced sections to the frame buffer, and then ending up with storing the rendered slices into a 3D texture. Depending on the rendering options, the result can be either surface or solid. The slice-based method benefits from hardware acceleration but suffers from incomplete boundary and missing thin region problems.

With the help of GPU, E. Eisemann et al. proposed a real-time voxelization approach[3][4]. In a later work, Zhao Dong et al.[2] presented another computational efficient voxelization algorithm for complex polygonal models by exploiting GPUs. They first convert the model into three discrete voxel spaces according to its surface orientation. The resultant voxels are encoded as 2D textures and stored in three intermediate sheet buffers called directional sheet buffers. These buffers are finally synthesized into one worksheet, which records the volumetric representation of the target. The whole algorithm traverses the geometric model only once and is accomplished entirely in GPU.

In addition, another branch of voxelization approaches are based on layer depth images generated by ray-tracing. Another data-parallel approaches for conservative and tile-based voxelizations are proposed by [19] in which both surface and solid voxelization methods are discussed. In addition, they proposed an octree-based sparse representation of the voxel sets for better utilization of memory resources, which enables higher voxelization resolution over 4096× 4096× 4096.

Currently, most voxelization techniques proposed so far only accepts closed or watertight objects as their input. For the non-manifold meshes or those with intersecting parts, the aforementioned methods cannot be applied to derive complete and correct volume data. To cope with such needs, Fakir S. Noorudin et. al. [17] proposed an election-based voxelization technique for the non-manifold meshes.

To deal with the cracks and the boundaries of the input mesh, they proposed a parity count method that begins with scan conversions by orthographic

projections in 13 directions, and then decided the final volume data by a majority vote. For interpenetrating subparts, they proposed a ray stabbing method that keeps only the first depth and the last depth samples for each ray. Consequently, a voxel is considered as an interior voxel in the final volume data only when the scan conversions in all projections classify the voxel as interior. Since these two techniques are not compatible with each other, when an input mesh contains both cracks/boundaries and interpenetrating subparts, they needs an additional repair process to seal the cracks in the first place, and then applies the ray-stabbing method afterwards to get the final volume data.

4. Our Robust Voxelization Approach

As we have mentioned earlier, previous works mostly constrained the input to be a watertight mesh. Only few provides solution for non-manifold meshes or those with interpenetrations. For models created for computer animation, a common way is to construct the model by means of a set of components[7][10]; more often, owing to some special needs such as cloth simulation[14], fabrication[18], and 3D scanning[20] etc., the model often consists of interpenetrating subparts. For these models, traditional solid voxelization methods mostly failed to find the interior voxels correctly, resulting in incomplete boundaries.

Furthermore, unwanted holes might be introduced in the interpenetrating regions with discontinuous volumetric data and defective visual appearance. A further use of such voxelization result, e.g., the skeletonization, is very likely to fail due to the defects. To solve this problem, we suggest a more robust voxelization approach depicted by the flowchart as shown in Fig. 2.

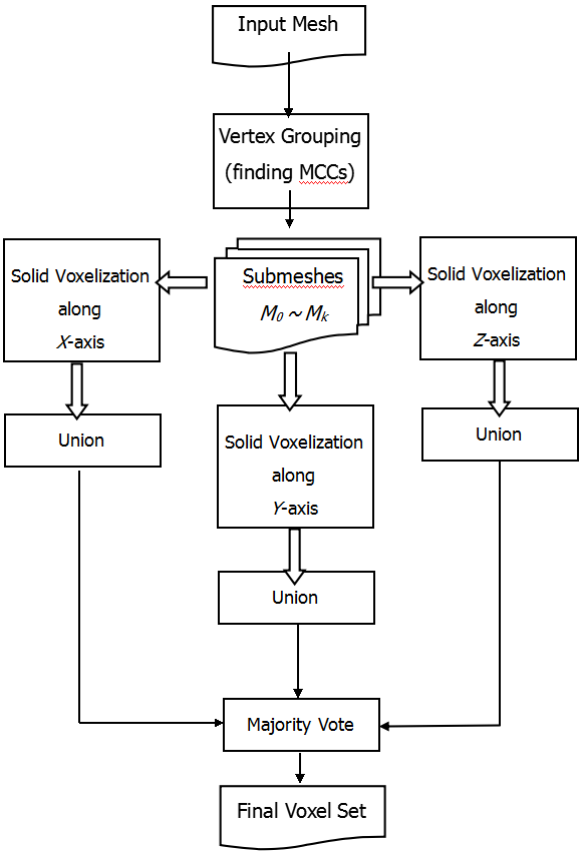


Figure 2: The flowchart of our robust voxelization approach.

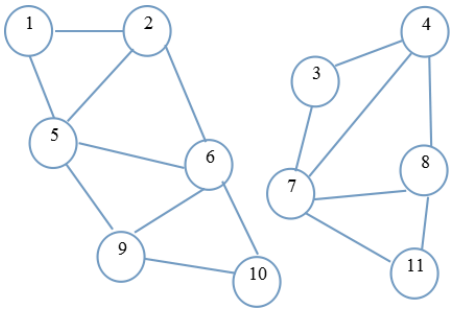
2.1 Vertex Grouping

The vertex grouping process is used to find connected components from the input mesh on the basis of the given topological relationship. It is equivalent to computing the connected components of a graph, which usually takes only linear time using either breadth-first search or depth-first search. In our approach, we begin with the assignment of a unique tag value to each vertex of the mesh representing its group ID followed by a merging process that iteratively replaces the tags of the vertices with the smallest tag value of their neighboring vertices. The

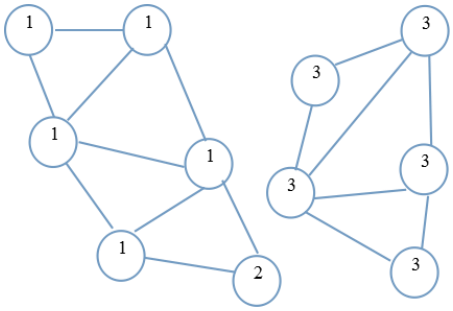
algorithm of vertex grouping is summarized as follows.

Require: $M(V,F)$, the input mesh;
1: **for all** $v_i \in V$; **do**
2: $G^v_{id}(v_i) = i$;
3: **end for**
4: **for all** $f_i \in F$; **do**
5: $G^f_{id}(f_i) = -1$;
6: **end for**
7: **repeat**
8: **update** = *false*;
9: **for all** $f_i = (v_a, v_b, v_c) \in F$, $v_a, v_b, v_c \in V$; **do**
10: $G^f_{id}(f_i) = \min\{G^v_{id}(v_a), G^v_{id}(v_b), G^v_{id}(v_c)\}$;
11: **if** $G^v_{id}(v_a) > G^f_{id}(f_i)$ **then**
12: $G^v_{id}(v_a) = G^f_{id}(f_i)$;
13: **update** = *true*;
14: **end if**
15: **if** $G^v_{id}(v_b) > G^f_{id}(f_i)$ **then**
16: $G^v_{id}(v_b) = G^f_{id}(f_i)$;
17: **update** = *true*;
18: **end if**
19: **if** $G^v_{id}(v_c) > G^f_{id}(f_i)$ **then**
20: $G^v_{id}(v_c) = G^f_{id}(f_i)$;
21: **update** = *true*;
22: **end if**
23: **end for**
24: **until update** = *false*;

In comparison with the traditional breadth-first and depth-first approach, our method performs fully in parallel by processing each vertex independently from the others. The merging stopped with no vertex has to alter its ID tag. An example illustrating such process is shown in Figs. 3(a)-(d).



(a)



(c)

Figure 3: The vertex grouping process. : (a) assigning vertex tag ID (different color represents different tags); (b)-(c) merging connected regions by replacing the tag of a vertex with the smallest tag value in its neighbouring vertices; (d) the final result.

After vertex grouping, the input mesh is subdivided into a number of maximally connected components. In the example of Al Capone mesh given in Fig. 4, twenty one maximally connected components are found from the original mesh.

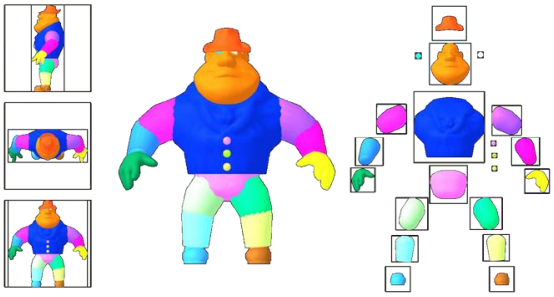


Figure 4: The twenty one maximally connected components from the Al Capone mesh found by the vertex grouping process.

2.2 Slice-based Voxelization

To voxelize the submeshes, we adopted a slice-based, or frame-based approach similar to Fang and Chen [5]. Initially, the model is aligned and normalized to a box volume later considered as the volume space of the model and its subparts. To prevent from the defects resulted by the incomplete boundaries and the thin regions, the voxelization is done individually to each subpart in the same volume space by a three-pass sliced-based solid voxelization process that converts the model along the directions of the three coordinate axes.

2.3 Voxels Synthesis

After the voxelizations of the subparts have completed, the voxel sets are then merged by a simple process as follows to get the final complete volumetric representation. Figures 5(a)-(c) illustrate the process of voxelization. Each vertex-group/subpart of the model is individually voxelized by scanning along the directions of the X, Y, and Z axis of the coordinate system.

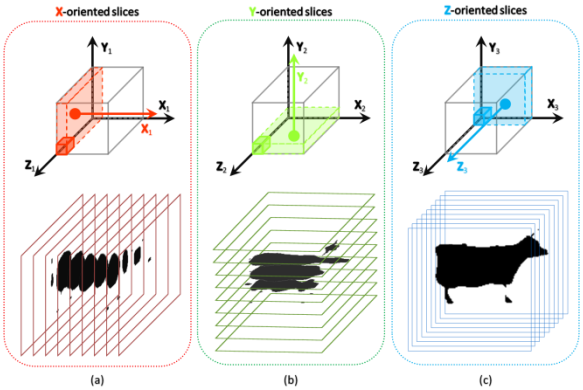


Figure 5: Adaptive solid voxelization in the direction of the (a) X-axis, (b) Y-axis, and (c) Z-axis.

Let the voxel sets derived by scanning a subpart Si of the input mesh M along the directions of the X, Y, and Z axis of the coordinate system be Si x,Si y,Si z, respectively. The final voxel set S is determined as follows.

Step 1. $\forall S_i$ of M, the voxel sets $S^x, S^y, and S^z$ derived along the directions of X, Y, and Z axes respectively are
1. $S^x = \cup_{\forall i} S_i^x$,
2. $S^y = \cup_{\forall i} S_i^y$,
3. $S^z = \cup_{\forall i} S_i^z$,

Step 2. The final voxe set S is then determined by performing majority vote over S^x, S^y, S^z.

With slice-based voxelization, the result obtained by scanning along a single direction may have the problems of incomplete boundary and missing thin regions[5]. For most cases, the result determined by performing majority vote over the directions of X, Y , and Z axes is sufficiently reliable.

5. Experimental Results

The experiments presented in this paper are performed on a PC equipped with Intel-Pentium Dual-Core CPU E5300 processor with 2GB RAM and an ATI Radeon HD 4550 powered graphics display card. In addition, all the programs are implemented with Microsoft Visual C++ 2008 running on Microsoft Windows 7.

Since our method emphasizes on its capability of dealing with non-manifold meshes and the meshes with interpenetrating parts, the test models we adopted in the experiments are either non-manifold or those with interpenetrations. Most of them are downloaded from the websites of the Aim@Shape and the Stanford 3D Scanning Repository. An example of such model has been given in Fig. 1, namely, the Al Capone mesh, comprising 21 inter-penetrating components.

To outline the differences between our approach and the traditional slice-based method, Fig. 6 shows the result derived from Fang and Chen's method[5].

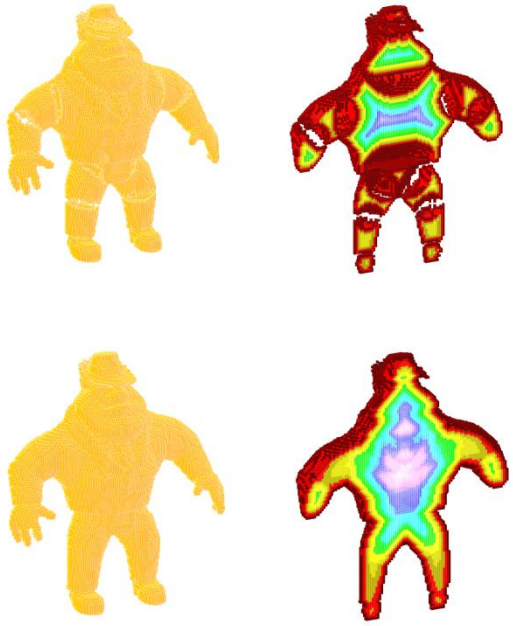


Figure 6: Upper row: the voxelization results by Fang and Chen's method[5]; lower row: the voxelization results by our method.

According to the results shown in Figure 6, the rendered image shows that a number of holes can be found at the interpenetrating areas. Furthermore, according to the sliced-view of the distance map, some parts of the voxel sets are discontinuous, which implies the presence of unwanted cracks. These problems are caused by the duplicated boundaries formed in the inter-penetrating regions in incomplete boundaries and holes. Such error-prone dataset may cause the follow-up applications such as the skeleton extraction deriving incorrect results. With our adaptive method, the result presented in Fig. 6 has no such problems.

To provide more empirical evidences, a number of additional test results on non-manifold meshes and those with interpenetrations are presented in Fig. 7.

Figure 7: The results obtained from our adaptive voxelization method for the Hand, Bunny, Dinopet, and Teapot meshes: left, the input mesh; middle, the voxel set; right, a sliced-view with coloured distance map.

The continuity of resulting volume data can be observed from a sliced-view of the coloured distance map along Z-axis presented in the right most column of Fig. 7; in which, the variation in colours of the image corresponds with the distance to the surface. For the non-manifold meshes, the Hand and Bunny, the final volume data are closed, continuous, and have no unwanted holes. For meshes comprising many interpenetrating subparts, i.e., the Dinopet and Teapot, the voxel sets are continuous and contain no unwanted holes. The numeric data from the experiments on the

meshes with interpenetrations are listed in Table 1. Since most of the calculations are performed by CPU, the execution times are relatively longer than those implemented on GPUs.

Table 1: The results of the meshes with interpenetrating parts

Model	V	F	Parts	Voxels	Grouping Time(Sec.)	Voxelization Time(Sec.)
Al Capone	3618	3440	21	108873	0.15	14.33
Teapot	3644	6320	4	203370	0.05	2.51
Dinopet	8145	15945	23	83769	0.23	18.60
Castle	6620	6493	21	149012	0.50	19.16

According to the results presented in Table 1, we may find that the total execution time is adversely influenced by the number of sub-parts, that means that a longer execution time is required by a model with more sub-parts.

6. Conclusions

In this paper we proposed a robust solid voxelization approach to a wild range of inputs including manifold, non-manifold meshes, and those with interpenetrations without using model repair and user intervention. According to the experimental results, the new method is capable of deriving robust continuous volume data of complete boundaries with no erroneous and redundant voxels, which is useful for a number of further applications such as the collision detection and skeleton extraction.

Furthermore, it is possible to use an alternative algorithm for voxelization. Provided that the efficiency is prominent, acceleration by parallel computing on GPU can be put into practice using GPU-based voxelization methods. Despite that our method is capable of processing meshes with cracks and interpenetrations for meshes with very thin shells or tinny features, the sampling resolution still has great impact on the correctness of voxelization.

References

[1]. Allard, J., Faure, F., Courtecuisse, H., Falipou, F., Duriez, C., Kry, P.G.: Volume contact constraints at arbitrary resolution. ACM Trans. Graph. 29(4), 82:1–82:10 (2010). DOI 10.1145/1778765.1778819. URL http://doi.acm.org/10.1145/1778765.1778819

[2]. Dong, Z., Chen, W., Bao, H., Zhang, H., Peng, Q.: Real-time voxelization for complex polygonal models. In: Computer Graphics and Applications, pp. 43–50 (2004). DOI 10.1109/PCCGA.2004.1348333

[3]. Eisemann, E., Decoret, X.: Fast scene voxelization and applications. In: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, I3D '06, pp. 71–78. ACM, New York, NY, USA (2006). DOI 10.1145/1111411.1111424. URL http://doi.acm.org/10.1145/1111411.1111424

[4]. Eisemann, E., Decoret, X.: Single-pass gpu solid voxelization for real-time applications. In: Proceedings of Graphics Interface 2008, GI '08, pp. 73–80. Canadian Information Processing Society, Toronto, Ont., Canada, Canada (2008). URL http://dl.acm.org/citation.cfm?id=1375714.1375728

[5]. Fang, S., Chen, H.: Hardware accelerated voxelization. Computers & Graphics 24(3), 433 – 442 (2000). DOI http://dx.doi.org/10.1016/S0097-8493(00)00038-8. URL http://www.sciencedirect.com/science/article/pii/S0097849300000388

[6]. Fang, S., Liao, D.: Fast csg voxelization by frame buffer pixel mapping. In: Proceedings of the 2000 IEEE Symposium on Volume Visualization, VVS '00, pp. 43–48. ACM, New York, NY, USA (2000). DOI 10.1145/353888.353896. URL http://doi.acm.org/10.1145/353888.353896

[7]. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Trans. Graph. 23(3), 652–663 (2004). DOI 10.1145/1015706.1015775. URL http://doi.acm.org/10.1145/1015706.1015775

[8]. Heidelberger, B., Teschner, M., Gross, M.H.: Real-time volumetric intersections of deforming objects. In: Proceedings of Vision, Modeling, and Visualization 2003, vol. 3, pp. 461–468. Akademische Verlagsgesellschaft Aka GmbH, Berlin, Germany (2003)

[9]. Hosssain, Z., Moller, T.: Edge aware anisotropic diffusion for 3d scalar data. IEEE Transactions on Visualization and Computer Graphics 16(6), 1376–1385 (2010). DOI 10.1109/TVCG.2010.147

[10]. Jain, A., Thormählen, T., Ritschel, T., Seidel, H.P.: Exploring shape variations by 3d-model decomposition and part-based recombination. Comput. Graph. Forum 31(2pt3), 631–640 (2012). DOI 10.1111/j.1467-8659.2012.03042.x. URL http://dx.doi.org/10.1111/j.1467-8659.2012.03042.x

[11]. Karabassi, E.A., Papaioannou, G., Theoharis, T.: A fast depth-buffer-based voxelization algorithm. Journal of Graphics Tools 4(4), 5–10 (1999). DOI 10.1080/10867651.1999.10487510. URL http://dx.doi.org/10.1080/10867651.1999.10487510

[12]. Kaufman, A., Cohen, D., Yagel, R.: Volume graphics. Computer 26(7), 51–64 (1993). DOI 10.1109/MC.1993.274942. URL http://dx.doi.org/10.1109/MC.1993.274942

[13]. Kaufman, A., Shimony, E.: 3d scan-conversion algorithms for voxel-based graphics. In: Proceedings of the 1986 Workshop on Interactive 3D Graphics, I3D '86, pp. 45–75. ACM, New York, NY, USA (1987). DOI 10.1145/319120.319126. URL http://doi.acm.org/10.1145/319120.319126

[14]. Li, J., Lu, G.: Modeling 3d garments by examples. Comput. Aided Des. 49, 28–41 (2014). DOI 10.1016/j.cad.2013.12.005. URL http://dx.doi.org/10.1016/j.cad.2013.12.005

[15]. Ma, C.M., Wan, S.Y.: Parallel thinning algorithms on 3d (18, 6) binary images. Computer Vision and Image Understanding 80(3), 364–378 (2000)

[16]. Miklos, B., Giesen, J., Pauly, M.: Discrete scale axis representations for 3d geometry. ACM Trans. Graph. 29(4), 101:1–101:10 (2010). DOI 10.1145/1778765.1778838. URL http://doi.acm.org/10.1145/1778765.1778838

[17]. Nooruddin, F.S., Turk, G.: Simplification and repair of polygonal models using volumetric techniques. IEEE Transactions on Visualization and Computer Graphics 9(2), 191–205 (2003). DOI 10.1109/TVCG.2003.1196006

[18]. Schulz, A., Shamir, A., Levin, D.I.W., Sitthi-amorn, P., Matusik, W.: Design and fabrication by example. ACM Trans. Graph. 33(4), 62:1–62:11 (2014). DOI 10.1145/2601097.2601127. URL http://doi.acm.org/10.1145/2601097.2601127

[19]. Schwarz, M., Seidel, H.P.: Fast parallel surface and solid voxelization on gpus. ACM Trans. Graph. 29(6), 179:1–179:10 (2010). DOI 10.1145/1882261.1866201. URL http://doi.acm.org/10.1145/1882261.1866201

[20]. Shen, C.H., Fu, H., Chen, K., Hu, S.M.: Structure recovery by part assembly. ACM Trans. Graph. 31(6), 180:1–180:11 (2012). DOI 10.1145/2366145.2366199. URL http://doi.acm.org/10.1145/2366145.2366199

[21]. Wang, Y.S., Lee, T.Y.: Curve-skeleton extraction using iterative least squares optimization. IEEE Transactions on Visualization and Computer Graphics 14(4), 926–936 (2008). DOI 10.1109/TVCG.2008.38