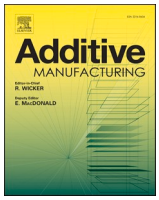
Additive Manufacturing 40 (2021) 101929

Contents lists available at ScienceDirect

Additive Manufacturing

journal homepage: www.elsevier.com/locate/addma



Research Paper

Direct 3D printing of multi-level voxel models

Sambit Ghadai, Anushrut Jignasu, Adarsh Krishnamurthy *

Department of Mechanical Engineering, Iowa State University, Ames, IA, USA

ARTICLE INFO

Keywords:
3D printing
Multi-level voxelization
GCode generation
Multi-level marching squares

ABSTRACT

We present a direct method for the additive manufacturing of multi-level voxelized models that achieves a better surface finish. We have developed a new multi-level marching squares algorithm to identify the boundary of the multi-level voxelized model. We have also developed methods to use the multi-level voxelization to perform the infill operation based on user-defined infill density. We directly generate the GCode that is input into the 3D printer for printing. Our method overcomes the issues associated with the slicing operation for standard CAD models. In addition, we can directly print thresholded voxel models that are output from CT or MRI scans to get a physical 3D representation of medical data. We show that our method performs well by directly printing test models of multi-level voxel representation of complex CAD geometries, and cardiac CT data.

1. Introduction

Additive manufacturing (AM, also known as 3D printing) is a process by which virtual CAD models are physically manufactured by adding material instead of traditional subtractive manufacturing. Most AM processes use tessellated CAD models in stereolithography (STL) or virtual reality modeling language (VRML) file format. Tessellated CAD models, in the form of triangle soups, can be easily obtained from generic CAD models, which are usually represented using boundary representation (B-rep) or constructive solid geometry (CSG) modeling of primitives. These tessellated models are further processed or *sliced* to obtain the layer-by-layer information for AM. However, other data structures representing solids, such as volumetric representations (voxels), point clouds, and medical imaging data (MRI and CT scans), require complex geometric algorithms to tessellate and represent them as triangular facets. Additionally, developing AM printing strategies for these representations is challenging due to the computational cost of processing these representations at higher fidelity. In this paper, we develop an AM strategy that enables direct 3D printing from volumetric representations or multi-level voxel models with higher accuracy and print quality. One of the advantages of using a voxel representation for AM is that the regularity of the voxel grid eliminates the need for explicit *slicing*, which is a time-consuming process. Slicing CAD models also creates gaps and discontinuities that need further processing to be used for AM. Voxelization alleviates this issue by directly generating a rectilinear occupancy grid conforming to the required layer structure. In addition, it also enables direct printing of models that are natively represented using voxels such as 3D imaging data from MRI and CT scan image stacks.

Voxel representations are rectilinear structured grids with scalar values representing the volume elements of a solid geometry. Capturing the fine details of a solid model requires a very high-resolution voxel grid. However, high-resolution uniform data structures for voxels require a large amount of memory and are compute-intensive. Specialized data structures such as octrees [9] and multi-level voxels [33] use a hierarchical approach to represent a dense voxel grid using sparser data. Specifically, multi-level voxels provide flexibility to choose the voxels' size and resolution at each hierarchical level. In addition, fast GPU-accelerated voxelization algorithms for multi-level voxels with very low space and time complexity have been developed [33]. We use a multi-level voxelization with two hierarchy levels for representing the volume information, without loss of generality. These levels–coarse and fine–have an user-defined selection of voxel grid resolution at each level. Specifically, we identify the voxels corresponding to the outside, inside, and the boundary of the object at the coarse voxel level using different scalar values. Each coarse boundary voxel is subdivided into fine voxels with the same scalar values representing the boundary, inside, and outside. The multi-level voxel representation is shown in Fig. 1, showing the difference with the regular voxel grid.

Multi-level voxel representation provides us with the necessary boundary and inside information to facilitate 3D printing. We build the surface boundary layers from fine voxels and compute the infill layers only at the coarse voxel level. The traditional marching squares algorithm is not suitable for a multi-level voxel grid as it can lead to errors in

* Corresponding author.
E-mail addresses: sambitg@iastate.edu (S. Ghadai), ajignasu@iastate.edu (A. Jignasu), adarsh@iastate.edu (A. Krishnamurthy).

https://doi.org/10.1016/j.addma.2021.101929

Available online 25 February 2021

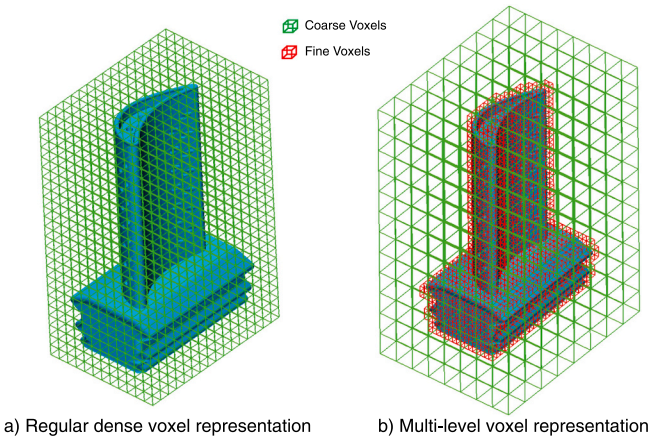


Fig. 1. Distinction between a regular voxel representation and multilevel voxel representation. (a) shows a regular voxel grid of a turbine CAD model while (b) shows its multi-level voxel representation. The effective representation resolution of both the grids are similar. (a) has a resolution of $12 \times 16 \times 24$ whereas (b) has a coarse resolution of $8 \times 8 \times 12$ and fine resolution of $4 \times 4 \times 4$, hence achieving effective resolution of $32 \times 32 \times 48$.

the final contour (see Section 3.1 for details). Therefore, we have developed a multi-level marching squares (MLMS) algorithm that generates high-resolution isocontours from multi-level voxel representation of CAD models to 3D print the boundary. However, printing the infill of a model with such higher accuracy is generally unnecessary as it does not contribute towards the surface finish. Hence, we use the inside information of fine and coarse voxels of the multi-level voxel representation to generate rectilinear infill structure using a hybrid scan-line approach. This approach conservatively extracts the infill lines closer to the boundary of a print layer while sparsely extracting the infill information from the larger coarse inside voxels at the same time. Thus, we achieve both high fidelity at the boundary and sparse infill structure with user-defined sparsity for direct 3D printing using multi-level voxels.

In addition to balancing the resolution of the isocontours and infill for 3D printing, multi-level voxel representation also allows us to print the infill with variable layer height without explicit slicing. This eliminates the staircase aliasing artifacts [28] on the boundary of the printed part, leading to a better surface finish. In the layer print direction, we print the isocontours based on each fine voxel layer but only print the infill once for the coarse voxel layer with a higher extrusion value that can be automatically set for the 3D printer's nozzle. Thus, we only print the infill once after printing all the isocontours of the fine voxels in the layer. We choose the resolution of the multi-level voxel representation to conform to the variable layer height parameters. The layer heights for boundary and infill structures govern the fine and coarse voxel sizes, respectively. We then voxelize the CAD model to this resolution using a GPU-accelerated multi-level voxelization algorithm. Based on these layer heights, we also generate the GCode instructions to command the printer to print the final part with variable heights, thus eliminating the need to slice the model.

In this paper, we have developed a method to directly 3D print a CAD model from a multi-level voxel representation. We have developed a multi-level marching squares algorithm, which can be used to 3D print medical data such as MRI and CT scans without explicit tessellation. The major improvements of our direct 3D printing approach over traditional STL 3D printing are tabulated in Table 1. Our main contributions include:

- A direct 3D printing method employing multi-level voxel representations of CAD models to accurately 3D print high resolution models.
- A multi-level marching squares algorithm to generate layer-by-layer contours from a multi-level voxel representation of a CAD model.
- A novel infill generation method that uses a hybrid scan-line algorithm to create variable height infill structures from multi-level voxel representations.
- Application of this 3D printing method to fabricate physical models from high resolution CAD and volumetric medical data such as stacks of MRI and CT scan images.

Table 1
Comparison of 3D printing methods.

Method	Implicit slicing	CT/MRI scans	Volume information
Traditional STL 3D printing	X	X	X
Direct 3D printing of multi-level voxels	✓	✓	✓

A complete outline of our framework is shown schematically in Fig. 2. We first voxelize the CAD model according to the user-defined layer heights and resolution to create a multi-level voxel representation consisting of coarse and fine voxels, each with its specific boundary and inside voxels (shown in green in Fig. 2). We then implement the multi-level marching squares algorithm (Section 3) on the boundary voxels of both coarse and fine voxels to generate the boundary isocontours (shown in blue). Using the inside voxels of the coarse and fine levels, we generate the infill structure for the model using a hybrid scan-line approach (Section 4, shown in red). Finally, we combine the boundary isocontours and the infill structure to create the GCode instructions (Section 5) using the user-defined layer heights. We show some physical examples of the 3D printed models in Section 6.

2. Background and related work

Given the recent evolution of 3D Printing technology, there has been an increase in research focused on improving various AM processes [23]. Most generic AM processes use computer-aided design (CAD) representation of a part or model to generate special additive 3D printing directives for manufacturing. Fadel and Kirschman [10] mentions that among a plethora of CAD representation formats that have been used for AM, none have been as universal as stereolithography (STL) format due to its simplicity. An STL file approximates a CAD model using a tessellated (triangular) surface model and is the de-facto representation for 3D printing of CAD models. The primary input to an AM process is a GCode file generated from the CAD model after performing the slicing operation on the STL file. However, issues like truncation errors and approximation of curved surfaces by triangular facets in tessellated representations introduce some level of inaccuracy [20]. Additionally, Kumar and Dutta [20] mention problems such as inconsistent normals, topological degeneracy, self-intersections, and geometric degeneracy to be associated with the conversion to STL format.

The slicing operation is a necessary part of the layered additive manufacturing process planning, which has a major impact on the surface finish [18,22,35], build time [19], and mechanical properties [24] of the manufactured model. There are various slicing algorithms like *uniform slicing* [6,7,34], its variant *adaptive slicing* [34], and *direct slicing* [5]. Uniform slicing generates slices with constant layer thickness and has been widely adopted for different AM processes. However, uniform slicing of relatively large CAD models is computationally intensive, and the process is slow [6,7], with coarse surface features. Adaptive slicing uses machine capability and geometry [18,22,35] to determine slice thickness allowing for reduced build time with a superior surface finish. Zhou et al. [35] proposes the use of non-uniform cusp heights for higher slicing efficiency. In this paper, we perform AM operations using voxel representations of CAD models to overcome the aforementioned issues associated with using tessellated representations and explicit slicing of

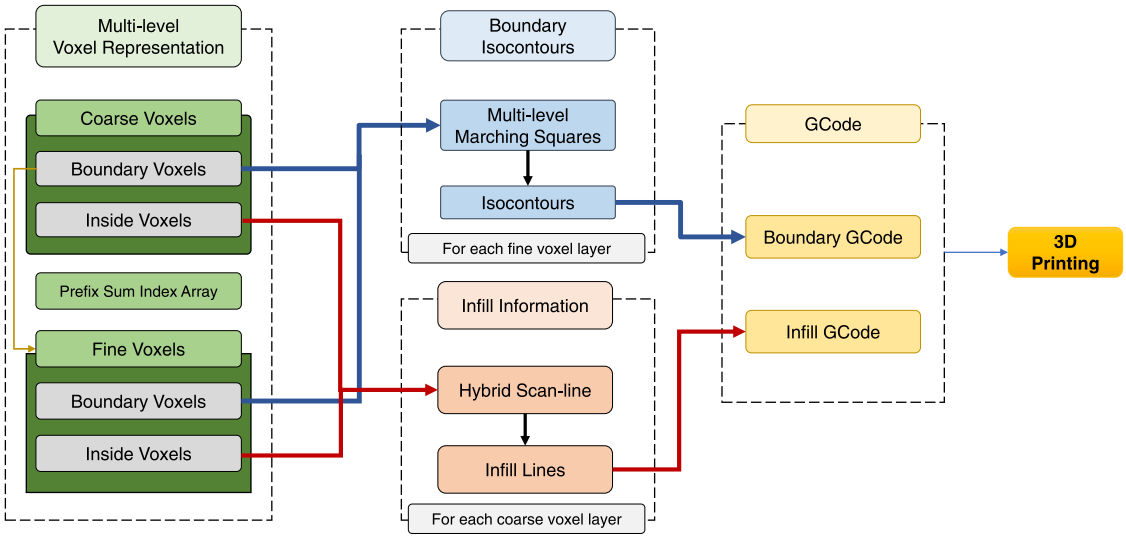


Fig. 2. Outline of direct 3D printing from multi-level voxels. Green boxes show detailed structure of multi-level voxel representation. Blue boxes define multi-level marching squares implementation on coarse and fine boundary voxels. Hybrid-scan line from coarse and fine inside voxels is shown in red. Final step of combining isocontours and infill to generated GCode is shown in yellow. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

triangular facets. Regular rectilinear voxel grids have implicit layer information that enables us to perform layered manufacturing directly without undergoing standalone slicing operations. In addition, we perform a variable layer height 3D printing for the surface as well as the interior of the model using our multi-level voxels.

Voxelization is a traditional volume representation schema for CAD models that stores the model's occupancy information in a regular 3D rectangular grid with volume elements called voxels. Voxelized representations offer the ability to conserve volumetric data associated with a CAD model [16] and allow for efficient Boolean operations and collision detection of CAD models. The conservation of volumetric data associated with a CAD model is of particular importance for 3D printing [3,11,16,29,30,32]. However, an accurate voxel representation of a CAD model requires a very large number of voxels (~1 billion) to represent the finer details of the model. This comes with a very high computational and memory cost.

To overcome this high memory cost, traditional volume representations have been modified to use various hierarchical data structures such as octrees and kD-trees. The Octree representation offers efficient memory allocation by using successive subdivisions of an object array into octants [14,25]. kD-trees, on the other hand, alternatively divide the space along the three principal axes, partitioning the space and allowing for memory-efficient handling of location queries [4]. Laine and Karras [21] use a sparse octree data structure where each node is represented as a voxel for ray casting. The sparse octree-based approach has also been used by Schwarz and Seidel [26] to create a GPU-based solid voxel representation while addressing the high memory consumption of traditional voxel grids. Kazhdan et al. [17] extract watertight level-set information from octrees. Similarly, Young and Krishnamurthy [33] uses the high computation capability of GPUs to generate a multi-level voxelization scheme that stores the occupancy information of a CAD model at two hierarchical levels of voxels. First, a general voxelization is performed on a boundary representation (B-rep) model to store the inside-outside and boundary occupancy information. Then, they further voxelize the boundary voxels into further smaller voxels and maintain the hierarchical relationship of both voxel levels using a prefix sum address data structure.

In this paper, we adopt the multi-level voxelization paradigm developed by Young and Krishnamurthy [33] to easily and rapidly voxelize models with an effective higher resolution, thus capturing the finer details of the CAD model. Further, multi-level voxel representation provides us with two distinct voxel hierarchy levels, which allows us to easily handle the boundary and infill generation for the AM process in a distinct fashion to achieve high surface printing accuracy while sparsely printing infill with a variable height and density.

3. Boundary extraction from a multi-level voxel model

In this section, we describe our methodology to directly print the boundary of the CAD model from the multi-level voxel representation. We first voxelize the B-rep solid model to create a multi-level voxel representation using the GPU-accelerated multi-level voxelization algorithm developed by Young and Krishnamurthy [33]. We voxelize a B-rep model since voxelization implicitly performs the slicing operation and generates the layer-by-layer information required for 3D printing. In addition, voxelization of the B-rep preserves the volume information of the CAD model in a voxel grid that enables us to generate the infill information for 3D printing. The multi-level voxelization can also be created from other 3D model representations such as point clouds and medical imaging data (see Section 6). We use two independent methods to extract and directly 3D print the boundary and the infill separately. The voxels having boundary information in each of the coarse and fine voxel resolutions, R_c and R_f respectively, are encoded using a scalar value *1.0* and the empty voxels are encoded using a scalar value *0*. Similarly, the voxels representing the inside of the CAD model are encoded with a scalar value *0.5*. The multi-level voxelization algorithm also creates a *prefix-sum* index array that maps each of the course boundary voxels to its specific fine voxels. Please refer to Young and Krishnamurthy [33] for more information about the multi-level voxel representation.

The coarse voxel resolution corresponding to the print direction is selected based on the required layer thickness and is independent of the voxel resolution in the lateral direction. The fine voxel resolution along the print direction is pre-selected and fixed according to the minimum printable layer height defined by the print nozzle of the 3D printer. Details of the voxel resolutions are further described in Section 5. To generate the 3D print directives for a voxelized CAD model boundary, we first isolate the layer information inherent in the voxel grid along the print direction. We then implement a marching squares based approach on each layer of the multi-level voxel grid to efficiently extract the isocontour information. We have developed a multi-level marching squares (MLMS) algorithm that extracts isocontours from the boundary

of multi-level voxel layers and combines them to generate the boundary printing information. Using MLMS on multi-level voxel grids allows us to create accurate isocontours from voxel grids with very high effective resolution (~ 8 billion voxels). Using the isocontours, we then generate the GCode for the 3D printer to print the boundary of the CAD model.

3.1. Multi-level marching squares

The multi-level marching squares algorithm takes in a single layer of multi-level voxels in a particular orthogonal slice of the voxel grid in the print direction as shown in Fig. 3. The hierarchical layers in a multi-level voxel model are shown in Fig. 4. In this model, we consider the Z direction of the voxel grid as the print direction and the X-Y plane as the print layer. We first extract a coarse plane or layer of coarse voxels L_c in the Z direction from the multi-level voxel grid. This coarse voxel layer further has multiple fine voxel layers defined by the fine voxel resolution R_f. We then extract a fine layer of voxel from L_c, namely L_f, that can be considered as a binary image with varying pixel resolution, as shown in Fig. 4(b). Since we have information regarding all the boundary voxels in the multi-level voxel grid, we only apply MLMS on the boundary voxels, i.e., voxels with a scalar value of *1.0*, thus exploiting the sparsity of a multi-level voxel grid. The boundary voxels on a multi-level voxel layer are hierarchically structured with each coarse boundary voxel B_c having further subdivision with fine boundary voxels B_f as shown in Fig. 3.

Performing the standard MS operation on a multi-level voxel grid is not feasible since the MS algorithm cannot automatically account for the heterogeneous neighboring voxels in a hierarchical multi-level voxel grid. Performing MS on the individual coarse voxel grid and combining the resulting iso-contours is also not feasible since it creates disconnected segments or gaps at each coarse voxel boundary, as shown in Fig. 5. MLMS overcomes these issues in multi-level voxel grids, or hierarchical data representation in general, to efficiently and accurately extract the iso-contours. To enable this, we pad each fine voxel grid, G_f, with a single line of fine voxels from its neighboring boundary voxel extremities. This is shown in Fig. 4(c) where we increase the grid size of G_f by augmenting the grid with its neighboring grid values at the extremes in $\pm x$ and $\pm y$ directions. This creates an overlapping grid between the fine boundary voxels of a voxel layer and ensures that the isocontour generated from standard MS on G_f is continuous along the object boundary. We perform the boundary augmentation based on the coarse voxel index of B_c to determine the neighboring voxels constituting the boundary.

Once we extract a fine layer of voxels L_f, we perform the standard marching squares (MS) algorithm individually on each of the coarse boundary voxels B_c that is in itself a voxel grid G_f consisting of fine voxels. Implementation of the standard marching squares algorithm involves visiting each of the fine voxels of G_g sequentially along the x and y direction (scanning or marching) with a filter of size 2×2. The filter compares the current scalar values of G_c with the standard marching squares look-up table of topological cases and draws an isocontour line intersecting the grid edges. The isocontour lines are drawn from the mid-points of the grid edges depending on the MS filter's topology. The MS algorithm creates a set of such isocontour lines per layer. It is important to note that each of these isocontour lines is directional, i.e., we preserve the sequence of the start and endpoints of the isocontour line. This later allows us to easily form a chain of such lines to create an isocontour for a particular profile in a voxel layer as described in Section 3.2.

3.2. Combined isocontour creation

Algorithm 1 shows all the steps of the MLMS algorithm. The standard MS algorithm on individual coarse boundary voxels B_c in a fine layer L_f creates a set of isocontour lines iso_f for that particular layer. Once the complete iso_f is generated for L_f, we check for duplicated isocontour lines in iso_f and remove the entries to have unique isocontour lines in the set. This is done to avoid adding multiple overlapping segments at each grid's extreme edge, which affects the 3D printing of the boundary. Duplicate isocontour lines are a result of augmenting G_f with its neighboring grid extremities due to every two adjacent grids sharing a single line of fine voxels, as shown in Fig. 4(c).

Algorithm 1. Multi-level marching squares algorithm.

Input: Multi-level voxel grid, G_m
Result: Multi-level Isocontours
foreach *Coarse Layer* $L_c \in G_m$ **do**
  foreach *Fine Layer* $L_f \in L_c$ **do**
    foreach *Coarse boundary voxel* $B_c \in L_f$ **do**
      Get fine voxel grid G_f from B_c
      foreach *Neighbor voxel of* B_c **do**
        if *Neighbor voxel is boundary* **then**
          Augment G_f extremes with 1
        else
          Augment G_f extremes with 0
        end
      end
      iso_f = MarchingSquares(G_f)
    end
    isocontours = JoinContours(iso_f, *hashmap*(iso_f))
  end
end

To combine the set iso_f into isocontours, we create a hash table that

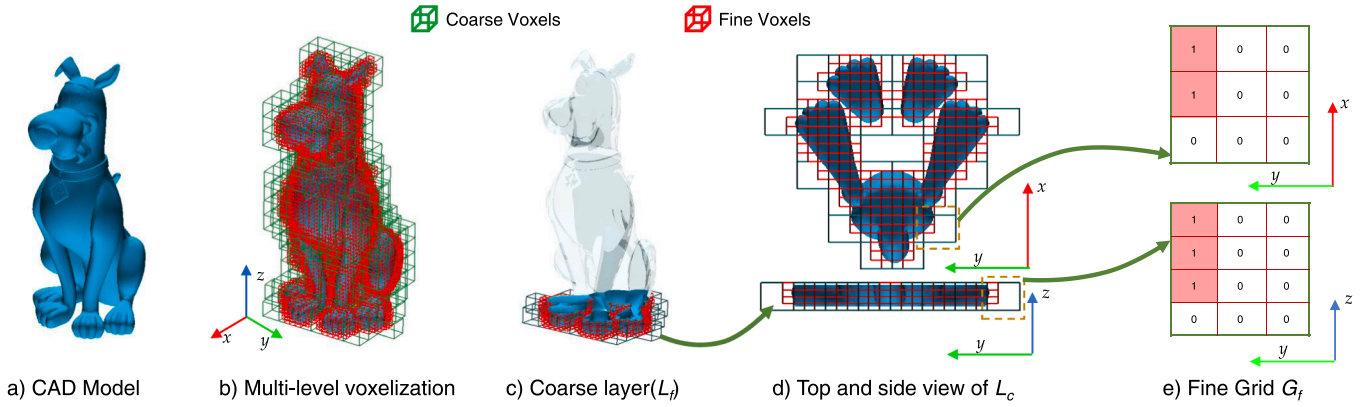


Fig. 3. In-depth view of a multi-level voxel grid. (a) shows a scooby CAD model which is voxelized to get multi-level voxelization as shown in (b). Green and red cubes represent the larger coarse voxels and smaller finer voxels at the boundary respectively. A single coarse voxel layer is extracted in (c) and its top and side views are shown in (d) with detailed fine voxels. (e) shows the fine voxel occupancy values of a single coarse voxel in the x-y and z directions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

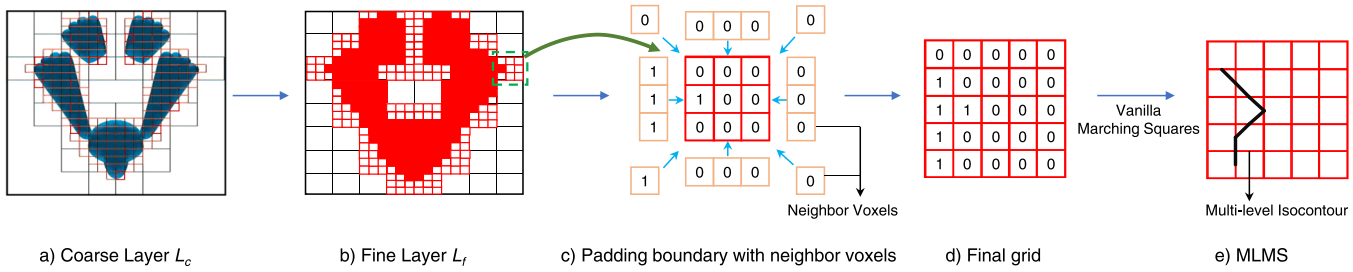


Fig. 4. Step-by-step implementation of Multi-level Marching Squares algorithm. (a) shows a coarse layer L_c in x-y direction. The fine boundary voxels are shown with a red fill in (b). (c) shows an example of coarse voxel with its boundary augmented with neighboring fine voxels and the final grid formed is shown in (d). Multi-level isocontour resulting from applying vanilla marching squares on the final grid is shown in (e). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

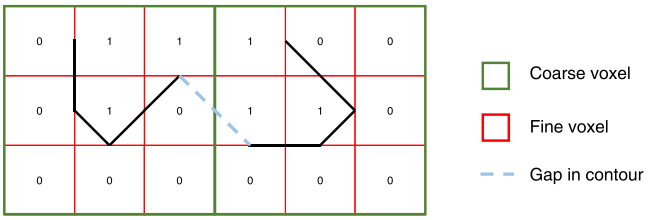


Fig. 5. Marching and gap issue while performing MS on multi-level voxel grid. The black lines are the isocontours generated using standard marching squares on the individual coarse voxels. The dashed line shows the gap in the isocontour between the two coarse voxels.

maps the endpoints of each isocontour line in iso_f. Since each mapping defines an isocontour line segment and each line segment has exactly two endpoints, the hash table mapping is bi-directional. Using this mapping, we then generate a chain of connected isocontour line segments, and a set of such isocontours form the boundary of the layer L_f. Fig. 6 shows the isocontours of a fine voxel layer extracted from the boundary voxels of a coarse voxel layer of the Scooby CAD model. We repeat the steps for generating isocontours from each L_f layer and further from each L_c layer, thus generating the isocontours for the whole model (shown in Fig. 7).

4. Infill generation for a multi-level voxel model

In this section, we present an infill 3D printing scheme for multi-level voxels. Several research works have recently focused on optimizing the infill pattern for efficiently printing a CAD model with better structural strength while using less material. Specifically, Aremu et al. [3] describe a topology optimization framework using voxels to determine the lattice

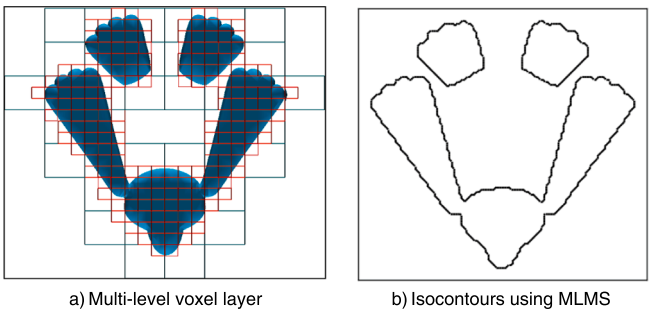


Fig. 6. Isocontours generated from a layer of Multi-level voxels using MLMS. (a) shows a layer with coarse and fine voxels as green and red boxes respectively. (b) shows the final isocontour generated from the multi-level voxel grid. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

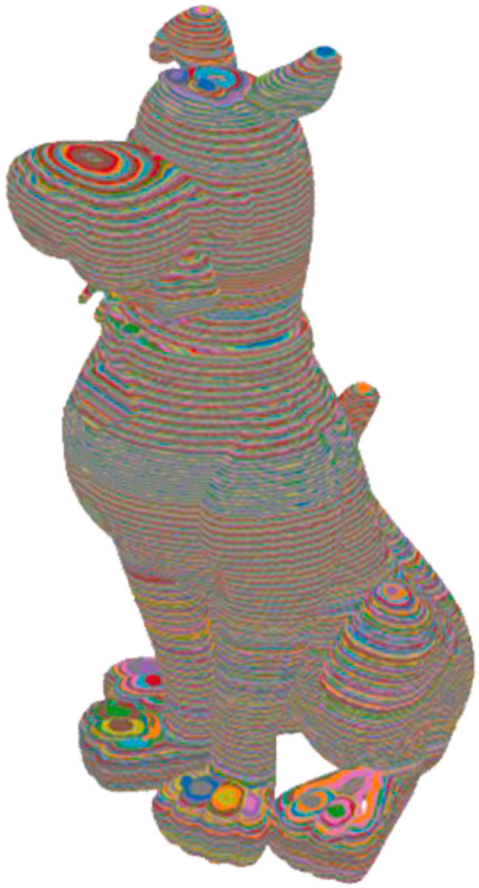


Fig. 7. Rendered view of all iso-contours generated from multi-level voxel representation of a scooby CAD model.

structure for 3D printing of infill. However, since this is not the main focus of our work, we describe a straightforward infill approach using a standard infill pattern. We describe an algorithm that makes efficient use of the multi-level voxel grids for this purpose. We use a rectilinear pattern, computed from the boundary of the multi-level voxel representation of the CAD model, to enable direct infill printing from voxels. However, generating the infill using a multi-level voxelization is not straightforward since the fine level at the boundary of the models needs to be accounted for during the infill generation process. Using our approach, we can still control the infill density directly, allowing for user-defined model density.

Algorithm 2. Hybrid scan-line algorithm on multi-level voxel grid.

```
Input: Multi-level voxel grid, G_m
Result: Infill Pattern
foreach Coarse Layer L_c ∈ G_m do
  foreach index_y ∈ L_c do
    foreach index_x ∈ L_c do
      Determine voxel: v(index_x, index_y)
      if v == 1.0 then
        Extract fine grid: G_f
        Fine layer L_f = TopLayer(G_f)
        foreach findex_x ∈ L_f do
          foreach findex_y ∈ L_f do
            Determine fine voxel: v_f(findex_x, findex_y)
            if v_f > 0 then
              Calculate center point, P
              AddPointToLine(P)
            end
          end
        end
      end
      if v == 0.5 then
        Calculate center point, P
        AddPointToLine(P)
      end
    end
  end
end
```

To compute the rectilinear infill pattern, we use a hybrid scan-line approach on both coarse voxel layers L_c and fine voxel layers L_f of the multi-level voxel grid using an adaptive technique. We first scan through L_c in one of the x or y directions until we encounter a coarse boundary voxel B_c, which in turn has a fine voxel grid G_f associated with it. Adapting to G_f, we scan through its top L_f layer only at the center voxel line. When the scanning encounters a fine boundary voxel B_f, we compute its center-point coordinates and append it to a data structure storing the continuous line segments of the scan-line. We then continue scanning along the direction and append points in B_f, where the scalar value of the voxels is > 0.5 (*inside* voxels). Once B_f is scanned, the scanning proceeds to the next coarse voxel and checks if it is a boundary or an inside voxel. The above steps are repeated in the case of a boundary voxel. However, if an inside voxel is scanned, the coarse voxel's center point is added to the scan-line data structure. This continues until another boundary voxel is encountered. In this case, the scanned line is extracted, and the scan is resumed in the same direction until it reaches the end of the multi-level voxel grid. A visual schematic of the hybrid scan-line algorithm on a multi-level voxel grid is shown in Fig. 8 and outlined in Algorithm 2.

One of the advantages of using this hybrid scan-line approach to determine the infill pattern is the accurate encapsulation of the infill by the boundary. Due to the presence of fine voxels at the coarse voxel grid boundary, the scan-line is accurately computed to match the object's boundary isocontour extracted in Section 3. Besides, this algorithm provides a control on the accuracy of the infill in z (layer) direction while enforcing the accuracy in x-y (layer plane) direction. In most additive manufacturing applications, the infill is required to be sparser than the surface boundary. We can easily control this using our algorithm by selecting the algorithm to scan every fine layer or a single fine layer in the multi-level voxel grid. We control sparse and dense infill pattern by selective scanning for inside voxels. For sparse infill, we scan a subset of the lines instead of all lines in the voxel grid. In addition, we

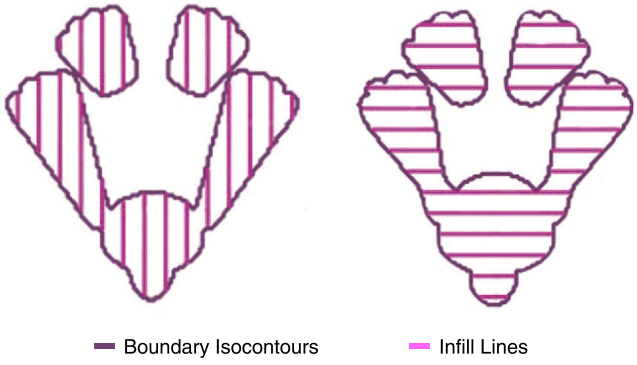


Fig. 9. Isocontour generation and alternating infill patterns in a print layer of scooby model. Figure on the left shows scanning in y direction and the right figure shows x direction scanning.

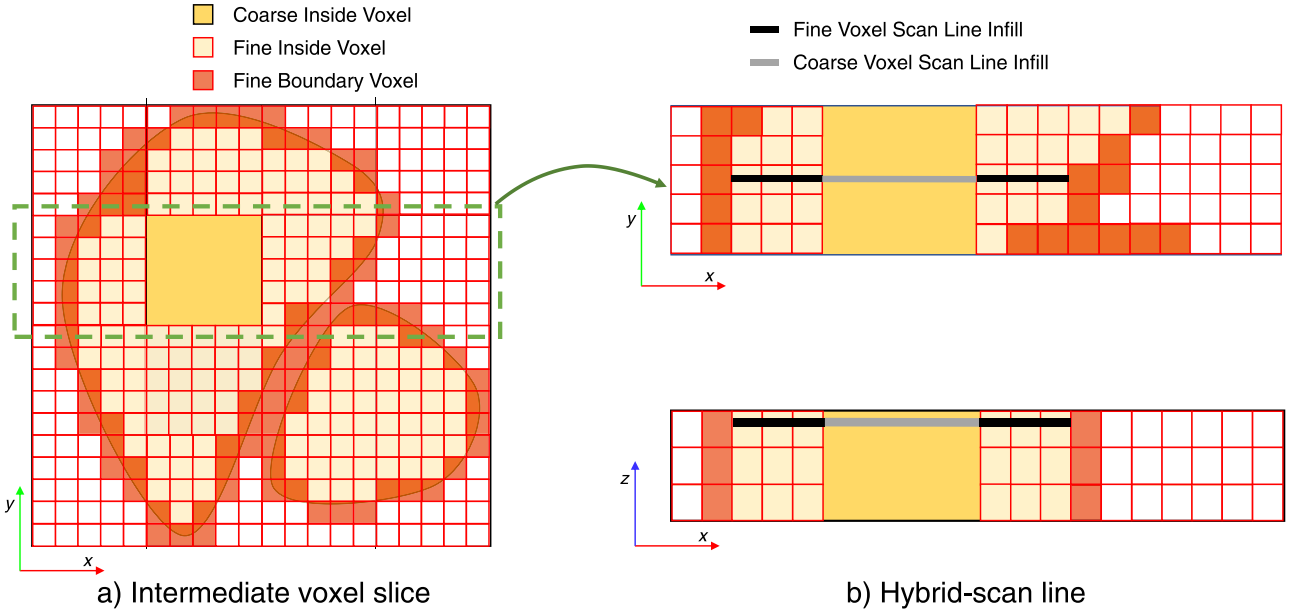


Fig. 8. Infill generation using hybrid scan-line in a multi-level voxel grid. An intermediate voxel layer is shown in (a) with coarse and fine voxels divided into inside and boundary voxels. A single scanning of a line of coarse voxels is represented by the green box. (b) shows the top and side views of the hybrid scan-line algorithm extracting infill line adaptively from fine and coarse inside voxels. Side view in (b) shows the layer height with which the extracted infill pattern is printed. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

change the direction of scanning for each infill layer by alternating between x and y directions. This is shown in Fig. 9 with both x and y scanning shown separately. After combining the layer, we generate a rectilinear infill pattern with the required density. Fig. 10 shows both sparse and dense rectilinear infill patterns generated from a layer of scooby model.

5. GCode generation

Once we compute the boundary isocontours and infill structures using the MLMS and the hybrid infill algorithm, we generate the GCode directives required to directly 3D print the multi-level voxel model. We implement a variable layer infill printing to enable efficient layer-by-layer printing by exploiting the embedded layer information from the multi-level voxels. Selecting an appropriate voxel resolution for the coarse and fine level voxels is the key to perform a variable layer infill and boundary printing. We develop the GCode directives for a Fused Deposition Modeling (FDM) 3D printer in this work. We note that we do not discuss the printing of support structures for overhangs in the model in this work. However, the multi-level voxel representation provides all the information required to generate such support structures–a possible future direction for this research.

In FDM 3D printing, one of the important properties that define a printed model's quality and rigidity is the nozzle diameter of the printer's extruder (hot end). Other properties, such as layer height (H_l), extrusion width (W_e), feed rate, traversal speed, and material consumption, are defined by the nozzle diameter [2,12,13]. In general, a printed model's layer height is selected to be within a range of the nozzle diameter for better print quality. Further, the infill pattern is designed to consume less material as it does not contribute to surface quality. We utilize these properties to selectively define the coarse voxel resolution as $N_{x1} \times N_{y1} \times H_l$ where N_{x1} is number of voxels in x direction, N_{x2} is number of voxels in y direction and layer height H_l is the number of voxels in z (print direction). We also define the fine voxel resolution as $N_{x2} \times N_{y2} \times 3$ where N_{x2} and N_{y2} are the number of fine voxels in x-y directions in a coarse boundary voxel. Hence, the effective resolution of our multi-resolution voxel grid is $(N_{x1}\dot{N}_{x2}) \times (N_{y1}\dot{N}_{y2}) \times (H_l\dot{3})$. In addition to providing a higher resolution in the x-y direction, we achieve a single layer height of $H_l/3$ due to the subdivision of the coarse voxels into three fine voxels in z direction.

We define the print directives for boundary isocontours generated in Section 3 with this layer height $H_l/3$, while the infill pattern generated in Section 4 has a height H_l. We select $H_l/3$ as the layer height (as shown in Fig. 11) since it enables us to capture finer details of the model. It allows a relatively small layer height to print the boundary with precision while having a large enough layer height to print the sparse infill structures without compromising the structural integrity. We print the infill layer of a higher thickness at each third layer of the boundary layer with this printing protocol.

To generate the GCode for FDM 3D printing, we compute the print material's extrusion values based on the layer heights, $H_l/3$ and H_l, and the distance between two consecutive points in the isocontours and infill patterns, respectively. This is shown in Fig. 11 with a thicker infill and thinner boundary layer extrusions. We aim to print compensate for printing lesser infill layers with a thicker infill according to the limits allowed by the 3D printer and the nozzle diameter. Printer specific instructions in the GCode, such as motor control, extruder temperature, external cooling fan control, and bed temperatures, are generic based on the printer and material used. We first compute the X, Y coordinates of the point in the isocontour to be printed and define the Z coordinated based on the layer index value of the respective isocontour. Further, we compute the extrusion value, E, and the feed rate F and define it in the GCode. Then we follow the process for all the isocontour points and profiles in the layer. This is repeated for all the L_f in L_c. After that, we proceed to follow the print process for the infill pattern of L_c with the appropriate E value computed from H_l. In addition, we swap the scan direction of the hybrid scan-line algorithm every consecutive layer to have alternating linear infill patterns for each layer. We follow this routine for all the L_c and L_f layers in the voxelized model to achieve the final GCode ready for printing.

6. Results

We printed different physical CAD models to demonstrate the direct 3D printing capability from multi-level voxels, including tessellated models and medical imaging data (CT scans). All the models were printed using the same plastic material (ABS) using the same 3D printer (*MakerGear M2*). We perform the GPU-accelerated multi-level voxelization of CAD models to get three separate data structures representing the coarse voxels, fine voxels, and a prefix sum address array that stores the fine voxel index values for each coarse boundary voxel (please see Young and Krishnamurthy [33] for details). We use the multi-level voxel information to get the GCode directives to print the model. The MLMS and the hybrid scan-line algorithms are implemented in a *Python* environment; the MLMS and GCode generation framework will be published open-source.

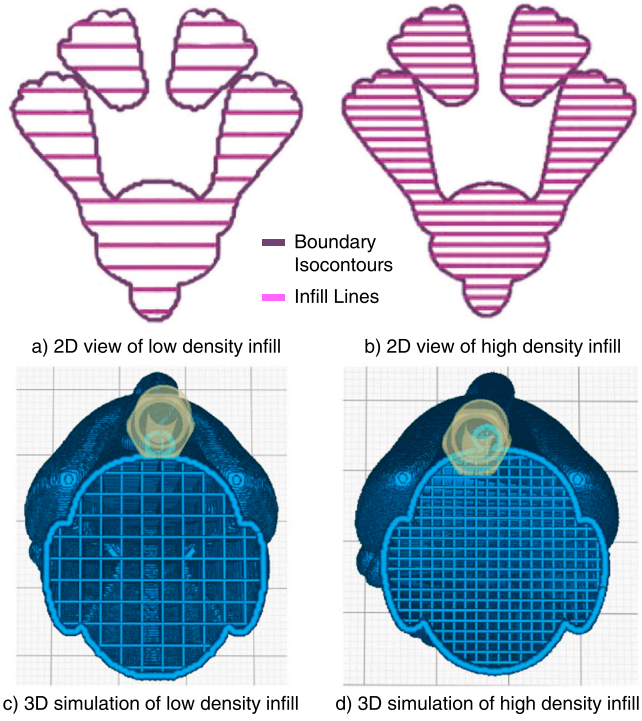


Fig. 10. Low and high density infill pattern visualizations in scooby model. (a) and (b) show the 2D view of the infill visualized from the GCode. (c) and (d) show sectional views of 3D printing simulation of the GCode. (a) and (c) are infill structures with low infill. (b) and (d) are infill structures with high density infill.

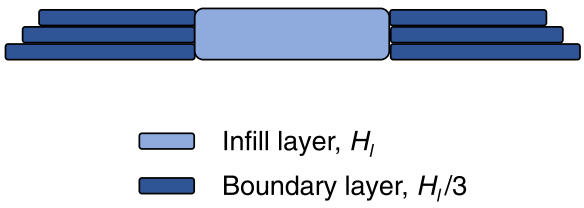


Fig. 11. Layer heights comparison of boundary layer and infill layer. Extrusion values of boundary and infill layers are calculated based on the respective layer heights. Fine voxel and coarse voxel resolution is also based on the layer heights defined for boundary and infill layers respectively.

To verify the printability using our framework, we simulated the 3D printing process using the generated GCode in the open-sourced Ultimaker Cura software [31]. Fig. 12 shows the final 3D printed simulation of a Scooby model which is voxelized to a coarse resolution of $64 \times 64 \times 512$ and fine resolution of $8 \times 8 \times 3$ thus producing an effective resolution of $512 \times 512 \times 1536$ voxels. In Fig. 12(b), the alternating pattern in the infill layer can be observed where each infill layer prints either along the horizontal or the vertical direction. Fig. 13 (b) further shows a cross-sectional view of the print simulation where there are two separate topological profiles of the model with a unique infill pattern strictly conforming to the respective boundaries. Similarly, Fig. 13 and Fig. 14 show the direct 3D printing simulations of the Stanford Bunny and a Turbine Blade model, respectively. Due to such a high-resolution representation of voxels, we can observe that in the *x-y* direction (print layer plane), the isocontours generated from fine voxels overcome the staircase effect that is generally associated with voxel grid isocontours. Specifically, in the z direction (print direction), the staircase effect is non-existent, and we obtain a smooth surface finish. Fig. 15 shows the Scooby, Stanford Bunny and Turbine Blade models, printed with resolutions of $512 \times 512 \times 1536$, $512 \times 512 \times 732$, and $512 \times 512 \times 876$, respectively using MLMS. Fig. 16 shows the final 3D printed Stanford bunny model printed from a low-resolution voxel grid of 64^3 voxel resolution. The voxel grid used in Fig. 16 is a standard 3D voxel grid, and the iso-contours and infill pattern is generated using standard marching squares and scan-line algorithms. In this figure, we observe that the staircase effect is obvious due to the low resolution of voxels. This issue is alleviated in Fig. 15 using high-resolution multi--level voxel grids along with MLMS and hybrid scan-line algorithms. Further, overhangs due to the absence of support structures are better handled in the high-resolution prints using multi-level voxel representation.

6.1. Direct printing of CT images

To enable direct 3D printing of CT images, we implemented our algorithm on a set of CT-scan images of the human heart. However, the CT images need to be preprocessed and converted to multi-level voxels before printing using our MLMS approach. The CT-scan images for the heart model are stacked on top of one another to create a voxel grid. The voxel grid has inside-outside occupancy information of the heart model as shown in Fig. 17(a) with a resolution of $512 \times 512 \times 116$. We create a multi-level voxel grid by performing a convolution operation on the original grid to create a coarse voxel grid (resolution of $64 \times 64 \times 116$) and a fine voxel grid (resolution of $8 \times 8 \times 4$). We copy each fine layer of voxels based on each image slice's thickness provided in the CT-scan data. This preserves the size and aspect ratio of the model, and the final multi-level voxel grid has an effective resolution of $512 \times 512 \times 464$, with 464 fine layers to be printed.

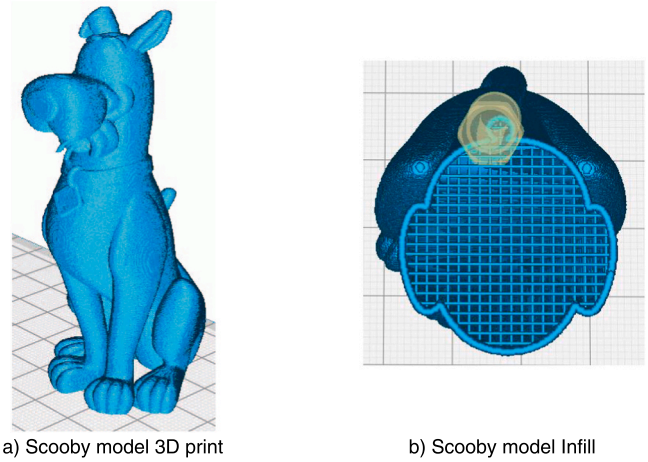


Fig. 12. 3D Print simulation for Scooby model. (a) shows the final 3D print simulation of the model with $64 \times 64 \times 512$ coarse and $8 \times 8 \times 3$ fine voxel resolutions. (b) shows cross-sectional view of the simulation with linear infill pattern.

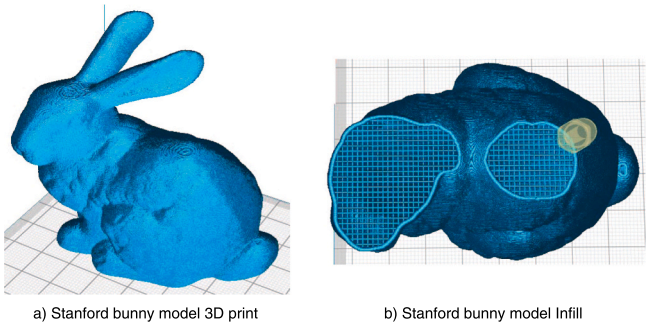


Fig. 13. 3D Print simulation for Stanford Bunny model. (a) shows the final 3D print simulation of the model with $64 \times 64 \times 244$ coarse and $8 \times 8 \times 3$ fine voxel resolutions. (b) shows sectional view of the simulation with linear infill pattern on two separate topological profiles of the model.

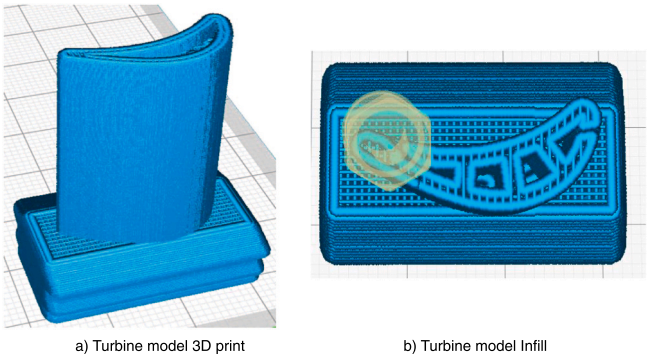


Fig. 14. 3D Print simulation for Turbine Blade model. (a) shows the final 3D print simulation of the model with $64 \times 64 \times 292$ coarse and $8 \times 8 \times 3$ fine voxel resolutions. (b) shows sectional view of the simulation with the infill pattern.

We use our direct 3D printing framework to generate the GCode information for the voxel grid created from CT images. Fig. 17(b) shows a direct 3D printing simulation obtained from a stack of CT-scan images of the human heart. Fig. 18 shows the final 3D printed heart model using our direct 3D printing method. Fig. 18(b) and 18(c) show the sectional views of the heart model with only the heart wall segmented from the CT-scan images. In Fig. 18, we observe that the surface quality of the 3D printed heart model is coarser along the z-direction than what we achieve from directly printing a CAD model using MLMS. This is due to the lower resolution of the original CT-image stack along the z-direction from which we extract the iso-contours for each layer. This lower-resolution results in a loss of accurate surface information in the voxel model of the CT-scan, which leads to a staircase effect at high curvature regions.

6.2. Print comparison

To validate our direct 3D printing framework, we compared different 3D printed models obtained using the traditional slicing method and our multi-level marching squares (MLMS) method. Fig. 19 shows the 3D prints of a simple curved geometry, in this case, a dumbbell model using a combination of primitive shapes. To show the correctness of the MLMS algorithm, we visualized the boundary of a single layer of the 3D print obtained using traditional slicing and MLMS methods (shown in

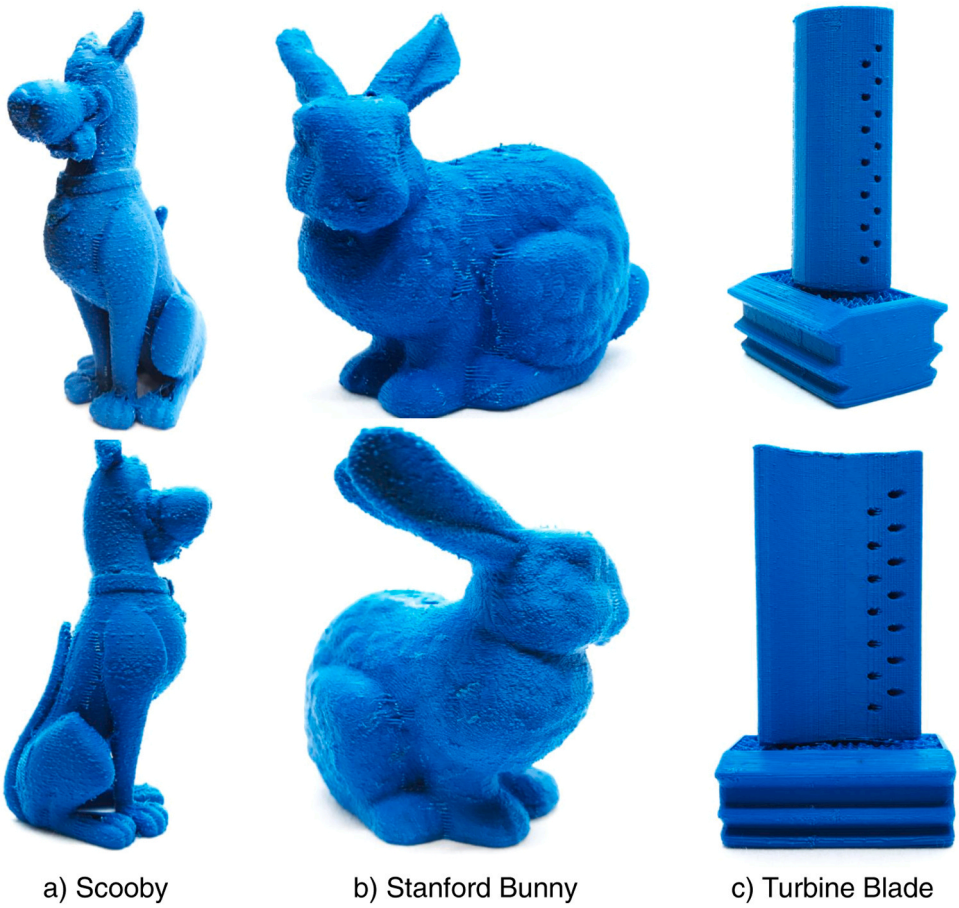


Fig. 15. High resolution ($> \mathbf{512^3}$) FDM 3D print using multi-level voxels. (a) shows the final 3D printed model of the Scooby, (b) shows the final 3D printed model of the Stanford Bunny, and (c) shows the Turbine Blade model.

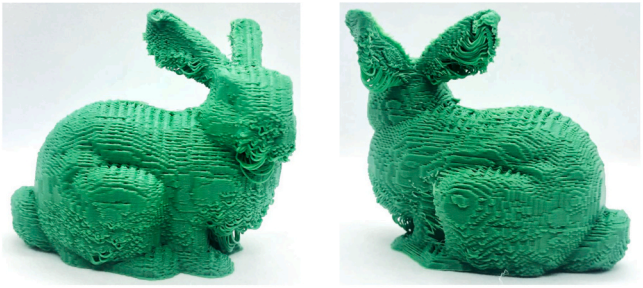


Fig. 16. Low resolution ($\mathbf{64^3}$) 3D print of the Stanford Bunny model using multi-level voxels. The staircase effect can be seen in this print due to the inability of the low-resolution voxel grid to capture the detailed features.

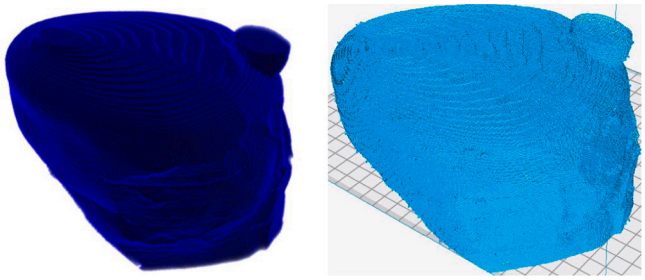


Fig. 17. 3D Print simulation of heart model from CT-scan image stack. (a) shows a volume rendering of the CT-scan images stacked on top of each other with a resolution of $64 \times 64 \times 116$ coarse voxels and $8 \times 8 \times 4$ fine voxels. (b) shows the 3D print simulation of the heart model.

Fig. 20). Comparing Fig. 20(a) and (b), it can be observed that the iso-contours of both the layers are identical in shape and smoothness.

In Fig. 21, we show the comparison of 3D printed models printed from the same 3D CAD model of the Stanford bunny using the traditional slicing method and our proposed direct 3D printing method from voxels. We observed that the surface quality of the model generated from our direct 3D printing method is mostly at par with the traditional slicing method and, at some regions, exceeds the quality of the slicing method generated model. In addition, the STL resolution artifacts are significantly reduced in our approach.

To quantify our print quality, we used 3D Systems Geomagic Capture 3D laser scanner [1] and Solidworks [27] to generate point cloud representations of the MLMS and STL printed versions of the Dumbbell and the Stanford Bunny model. The point clouds were exported to the Cloud Compare software [8]–an open-source framework for computing point cloud metrics. We then segmented the point clouds to generate patches of densely scanned points of a specific region of the printed model. Visualizations of the point cloud patches for the Dumbbell model are shown in Fig. 22, and the Stanford Bunny model are shown in Fig. 23. The patches were obtained approximately from the same regions of the MLMS, and STL printed models. Using fine registration, we aligned the individual patches with the CAD model. We then generated the error metrics between the CAD model and the MLMS model or traditionally sliced STL model patches using the point cloud-mesh-compare functionality of Cloud Compare. The quantitative error metrics obtained

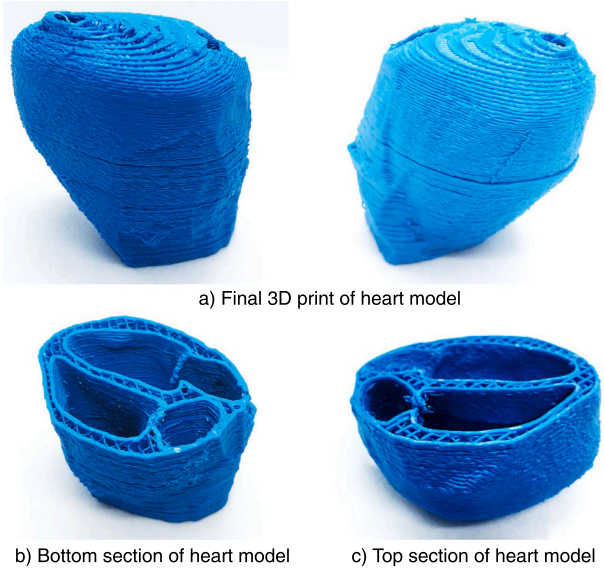


Fig. 18. 3D printed heart model from CT-scan. The CT-scan is segmented to generate a voxel grid containing only the walls of the heart model. (a) shows the complete model; (b) shows the bottom section of the heart model showing the four heart chamber cavities; and (c) shows the top section of the heart model.

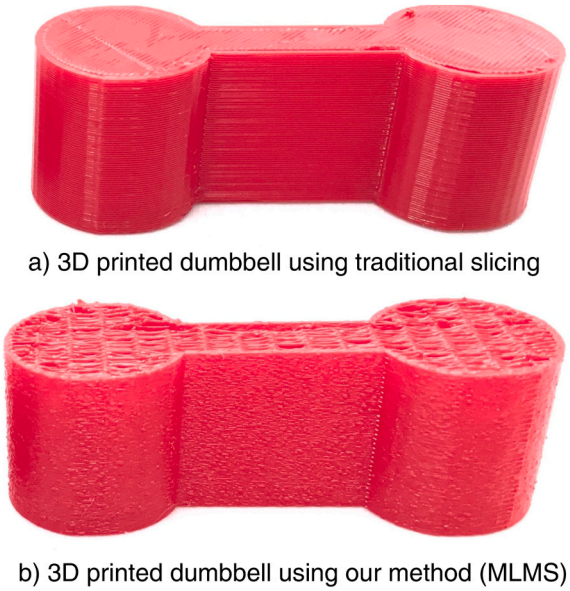


Fig. 19. 3D print comparison between the traditional slicing method and our direct 3D printing from multi-level voxels for a simple model.

from the point cloud comparisons from both models are tabulated in Table 2. It can be seen that, for both the Dumbbell and the Bunny models, the average distance between the CAD model and the point cloud obtained from MLMS 3D print is less than that of the point cloud obtained from STL based 3D print. Further, the maximum error between the MLMS point cloud patch and the CAD model is nearly identical for the Dumbbell model and lower in the Bunny model than the STL obtained point cloud. This quantitative analysis shows that our approach of direct 3D printing from multi-level voxel models has a better surface accuracy than traditional slicing based methods.

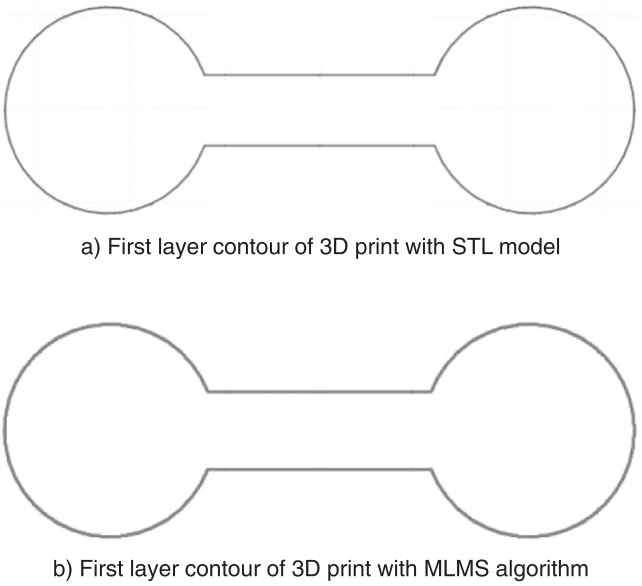


Fig. 20. Comparison of iso-contour extraction using the traditional slicing method and our MLMS algorithm. (a) shows the first layer contour of a simple dumbbell model printed using traditional slicing method (b) shows the first layer contour of a simple dumbbell model printed using direct 3D printing from multi-level voxels.

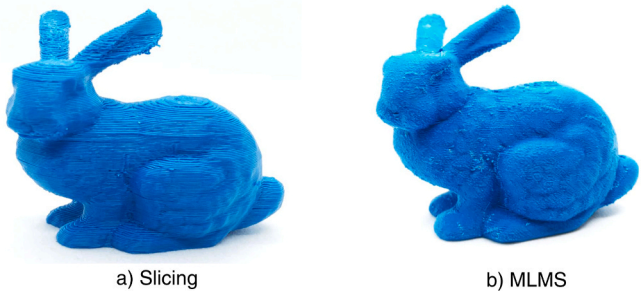


Fig. 21. 3D print comparison between the traditional slicing method and our direct 3D printing from multi-level voxels method. (a) shows the Stanford Bunny model printed using the traditional slicing method, and (b) shows the Stanford Bunny model printed using direct 3D printing from multi-level voxels.

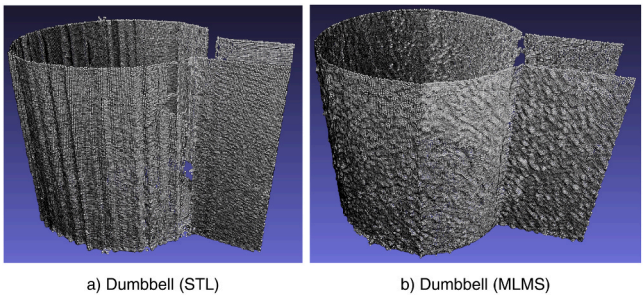
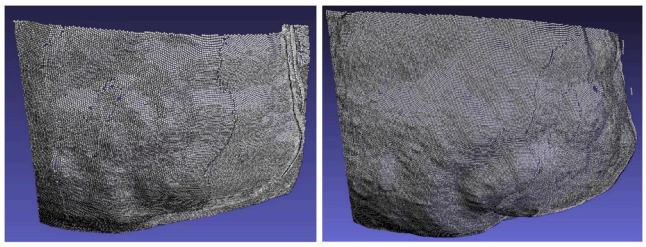


Fig. 22. Point cloud visualization of patches extracted from 3D printed Dumbbell models. (a) shows the point cloud patch of the model printed using the traditional slicing method, and (b) using multi-level voxels.

7. Discussion, limitations & future work

We discuss a few observations as well as the limitations of the current work in this section. We did not explore support structure generation for overhang sections in a CAD model. This can be seen in the final 3D



a) Stanford Bunny (STL) b) Stanford Bunny (MLMS)

Fig. 23. Point cloud visualization of patches extracted from 3D printed Bunny models. (a) shows the point cloud patch of the model printed using the traditional slicing method, and (b) using multi-level voxels.

Table 2
Quantitative comparison of 3D printed models with input CAD models, using traditional slicing method (STL) and our multi-level marching squares (MLMS) method.

Error metric (mm)	Dumbbell		Stanford Bunny	
	STL	MLMS	STL	MLMS
Maximum Error	0.2663	0.2667	0.1574	0.1476
Average Distance	0.1276	0.0046	0.0068	0.0050
Standard Deviation	0.2371	0.2730	0.4669	0.2790

printed objects, where due to the absence of support structures, we get warped boundaries at overhang locations of the model. However, due to the high-resolution voxelization using multi-level voxel representation, the generated contours are highly accurate, which allows us to 3D print the complete models without incurring a hefty penalty on the surface quality of the overhangs.

We observed that regions with steep angles resulted in a more prominent staircase effect in the final 3D print. This is due to the discretization of the surface during the voxelization. Regions with steep angles result in a significant loss of surface continuity after voxelization. Extracting iso-contours of such regions using the MLMS algorithm reduces the staircase effect. However, high curvature surfaces are still prone to this effect due to the intrinsic nature of the marching squares algorithm, where the voxel center points are considered for iso-contour extraction. This can be observed in Fig. 18 at the top region of the heart model. Using a more sophisticated contour generation algorithm, such as Dual Contouring [15] might overcome this limitation.

As presented in Section 6, high-resolution voxel grids enable us to print 3D models with higher accuracy and better surface finish. However, iso-contours extracted from very high-resolution voxel grids ($> 1024^3$) result in any two adjacent points being very close (order of $> 10^{-4}$ mm) to each other that is less than the precision of the 3D printer extrusion. Due to this limitation, the extrusion of print material from the nozzle becomes inconsistent, introducing a discontinuity in the material flow, thus affecting the surface finish of the printed model. Hence, the voxel resolution of the 3D model is still limited by the precision and accuracy of the 3D printer.

The multi-level voxel representation is not readily available for medical data. We pre-processed the high-resolution medical data to generate a multi-level voxel representation that is true to the original data in the *x-y* layer plane. However, due to the stacking up of a copy of each image slice in the print direction, there is a noticeable staircase effect in the z-direction. This is primarily due to the limitation of the data and not of the method itself. We can easily overcome this limitation with the availability of a high-resolution CT scan stack.

Future work for the direct 3D printing framework involves developing additional methods to generate support structures for overhangs. Along with the boundary and inside-outside information for the voxel grid, we also store the average triangle normal information for each voxel based on the tessellated CAD model. This normal information can be used to find overhang regions in the model and generate appropriate support structures. In addition, different infill structure patterns can be explored to be 3D printed using voxel grids. A potential improvement to the framework is also to generate multiple offsets of the iso-contour in a layer to create a boundary *skin* in the 3D printed model. This will improve the surface quality of the prints by reducing the effect of over-extrusion of infill over the boundary, thus eliminating small blobs on the surface.

8. Conclusions

In this paper, we have developed an additive manufacturing framework to directly 3D print CAD models from multi-level voxel representation. The multi-level voxel representation has inherent layer information, obviating the need for the slicing operation. We extract accurate iso-contours from high-resolution multi-level voxels using our multi-level marching squares algorithm to print the model's boundary surface. This method exploits the sparse nature of the multi-level voxel representation to heavily reduce the aliasing or staircase effect in the model, achieving a better surface finish. We generate and print the infill structure of the voxel model with a height proportional to the coarse voxels while the boundary iso-contours are printed with layer height proportional to the fine voxels, which further reduces the aliasing effects in the z-direction. In addition, we can also directly control the infill density, leading to prints with user-defined weight. We believe this approach will be widely adopted by the additive manufacturing community due to its flexibility and ease of adapting it to different layered manufacturing processes.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Science Foundation, USA under grant numbers CMMI-1644441 and OAC-1750865.

Appendix A. Supporting information

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.addma.2021.101929.

References

[1] 3DS Systems Inc., Geomagic Capture Series 3D Scanner. ⟨https://www.3dsystems.com/3d-scanners/geomagic-capture⟩.
[2] A. Alafaghani, A. Qattawi, B. Alrawi, A. Guzman, Experimental optimization of fused deposition modelling processing parameters: a design-for-manufacturing approach, Procedia Manuf. 10 (2017) 791–803.
[3] A.O. Aremu, J.P.J. Brennan-Craddock, A. Panesar, I.A. Ashcroft, R.J.M. Hague, R. D. Wildman, C. Tuck, A voxel-based method of constructing and skinning conformal and functionally graded lattice structures suitable for additive manufacturing, Addit. Manuf. 13 (2017) 1–13.
[4] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (1975) 509–517.
[5] X. Chen, C. Wang, X. Ye, Y. Xiao, S. Huang, Direct slicing from PowerSHAPE models for rapid prototyping, Int. J. Adv. Manuf. Technol. 17 (2001) 543–547.
[6] S.H. Choi, K.T. Kwok, A memory efficient slicing algorithm for large STL files, in: Proceedings of the 1999 International Solid Freeform Fabrication Symposium, (1999) pp. 155–162.
[7] S.H. Choi, K.T. Kwok, Hierarchical slice contours for layered-manufacturing, Comput. Ind. 48 (2002) 219–239.
[8] CloudCompare, Cloudcompare (version 2.11.3), (2020). ⟨http://www.cloudcompare.org/⟩.
[9] C. Crassin, S. Green, Octree-Based Sparse Voxelization Using the GPU Hardware Rasterizer, CRC Press, 2012, pp. 303–319.
[10] G.M. Fadel, C. Kirschman, Accuracy issues in CAD to RP translations, Rapid Prototyp. J. (1996).

[11] S. Ghadai, A. Balu, S. Sarkar, A. Krishnamurthy, Learning localized features in 3D CAD models for manufacturability analysis of drilled holes, Comput. Aided Geom. Des. 62 (2018) 263–275.
[12] J. Giri, A. Patil, H. Prabhu, The effect of various parameters on the nozzle diameter and 3D printed product in fused deposition modelling: an approach, in: C. R. Krishna, M. Dutta, R. Kumar (Eds.), Proceedings of 2nd International Conference on Communication, Computing and Networking, Springer Singapore, Singapore, 2019, pp. 839–847.
[13] G. Gomez-Gras, R. Jerez-Mesa, J.A. Travieso-Rodriguez, J. Lluma-Fuentes, Fatigue performance of fused filament fabrication PLA specimens, Mater. Des. 140 (2018) 278–285.
[14] C.L. Jackins, S.L. Tanimoto, Octrees and their use in representing three-dimensional objects, Comput. Graph. Image Process. 14 (1980) 249–270.
[15] T. Ju, F. Losasso, S. Schaefer, J. Warren, Dual contouring of hermite data, in: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, (2002) pp. 339–346.
[16] A. Kaufman, Voxels as a computational representation of geometry, The computational representation of geometry. SIGGRAPH 94, (1994) 45.
[17] M. Kazhdan, A. Klein, K. Dalal, H. Hoppe, Unconstrained isosurface extraction on arbitrary octrees, in: Symposium on Geometry Processing, (2007) pp. 1–9.
[18] P. Kulkarni, D. Dutta, An accurate slicing procedure for layered manufacturing, Comput. Aided Des. 28 (1996) 683–697.
[19] P. Kulkarni, A. Marsan, D. Dutta, A review of process planning techniques in layered manufacturing, Rapid Prototyp. J. (2000).
[20] V. Kumar, D. Dutta, An assessment of data formats for layered manufacturing, Adv. Eng. Softw. 28 (1997) 151–164.
[21] S. Laine, T. Karras, Efficient sparse voxel octrees, IEEE Trans. Vis. Comput. Graph. 17 (2010) 1048–1059.
[22] W. Ma, W.C. But, P. He, NURBS-based adaptive slicing for efficient rapid prototyping, Comput. Aided Des. 36 (2004) 1309–1325.
[23] E. Matias, B. Rao, 3D printing: on its historical evolution and the implications for business, in: Proceedings of the 2015 Portland International Conference on Management of Engineering and Technology (PICMET), IEEE, (2015) pp. 551–558.
[24] D. Popescu, A. Zapciu, C. Amza, F. Baciu, R. Marinescu, FDM process parameters influence over the mechanical properties of polymer specimens: a review, Polym. Test. 69 (2018) 157–166.
[25] H. Samet, An overview of quadtrees, octrees, and related hierarchical data structures. Theoretical Foundations of Computer Graphics and CAD, Springer, 1988, pp. 51–68.
[26] M. Schwarz, H.P. Seidel, Fast parallel surface and solid voxelization on GPUs, ACM Trans. Graph. (TOG) 29 (2010) 1–10.
[27] SolidWorks Corp., SolidWorks, (2018). ⟨www.solidworks.com⟩.
[28] H.C. Song, N. Ray, D. Sokolov, S. Lefebvre, Anti-aliasing for fused filament deposition, Comput. Aided Des. 89 (2017) 25–34.
[29] P. Song, Z. Fu, L. Liu, C.W. Fu, Printing 3D objects with interlocking parts, Comput. Aided Geom. Des. 35 (2015) 137–148.
[30] A. Telea, A. Jalba, Voxel-based assessment of printability of 3D shapes, in: Proceedings of the International symposium on mathematical morphology and its applications to signal and image processing, Springer, (2011) pp. 393–404.
[31] Ultimaker, 2011–2020. Ultimaker cura. ⟨https://ultimaker.com/software/ultimaker-cura⟩.
[32] Y. Xie, X. Chen, Support-free interior carving for 3D printing, Vis. Inform. 1 (2017) 9–15.
[33] G. Young, A. Krishnamurthy, GPU-accelerated generation and rendering of multi-level voxel representations of solid models, Comput. Graph. 75 (2018) 11–24.
[34] Z. Zhang, S. Joshi, An improved slicing algorithm with efficient contour construction using STL files, Int. J. Adv. Manuf. Technol. 80 (2015) 1347–1362.
[35] M.Y. Zhou, J.T. Xi, J.Q. Yan, Adaptive direct slicing with non-uniform cusp heights for rapid prototyping, Int. J. Adv. Manuf. Technol. 23 (2004) 20–27.