Support-Free Interior Carving for 3D Printing

Abstract

Recent interior carving methods for functional design necessitate a cumbersome cut-and-glue process in fabrication. We propose a method to generate interior voids which not only satisfy the functional purposes but are also support-free during 3D printing. We design a support-free unit structure for voxelization and derive the wall thicknesses parameterization for continuous optimization. We also design a discrete dithering algorithm to ensure the printability of ghost voxels. The interior voids are iteratively carved by alternating the optimization and dithering. We apply our method to optimize the static and rotational stability, and print various results to evaluate the efficacy.

(a) Printing (b) Interior voids (c) Result

Figure 1: ***Support-free interior carving.*** *(a) A close-up of the* HORSE *during printing, without any supporting structures inside it. (b) A sectional view of the support-free interior voids. (c) The* HORSE *printed as a whole can stand stably.*

1 Introduction

Interior shape carving is a modeling operation to hollow voids inside an object without affecting its exterior appearance. In real-world fabrication, the operation is essential for modifying physical properties, such as weight, center of mass and moment of inertia, of an object to achieve its functional purposes. Despite its usefulness, manual hollowing is a tedious trial-and-error process even for a simple task. To overcome such difficulties, recent work has investigated several computational methods to hollow digital models automatically. By using these methods and increasingly popular 3D printers, a personal user can handily design and fabricate functional objects that stand on ground [Prévost et al. 2013; Christiansen et al. 2015], spin around an axis [Bächer et al. 2014], float in fluid [Wang and Whiting 2016] and withstand under load [Lu et al. 2014].

The mainstream 3D printing technologies, such as fused deposition modeling (FDM) and stereolithography (SLA), deposit materials layer-by-layer to build a tangible product. During printing, additional supporting structures are often necessary to avoid the falling of overhanging parts, e.g. to support the ceilings of interior voids. The supporting structures affect the computed physical properties thus must be removed from the printed object. However, this process usually leaves visually unpleasant cracks on the object surface [Zhang et al. 2015b; Wang et al. 2016]. What is even worse is that supporting materials inside interior voids of the object cannot be directly taken out. Previous work tackles this issue by first cutting the model into individual parts to print and then gluing printed pieces back together, which is a cumbersome process.

Thanks to the properties of plastic materials like ABS and PLA, an FDM printer can build slanted walls without using support materials. Inspired by this observation, we realize that it is possible to make the interior voids support-free, by constraining the structures and boundary slopes (Figure 1). An object with such interior voids can be printed as a whole, eliminating the needs of any supporting structures and shape decompositions.

In this paper, we propose a novel method to automatically generate support-free interior voids while satisfying user-specified functions. We design a voxel structure that is self-supportable during printing and use it as the basic construction unit of interior voids. Taking a step further, we incorporate the support-free voxel structure into an interior carving optimization framework to achieve functional design. Specifically, we use wall thickness to parameterize the integral terms over each voxel and use them to formulate the objective function of the optimization problem. To deal with the ghost voxels generated in the optimization, we develop an efficient dithering algorithm that ensures printability and preserves support-free property. The interior voids are iteratively carved by alternating between an application specific nonlinear programming and the ghost voxels dithering. We apply the method to design statically balanced objects and spinnable objects, and print all the results without using any support materials inside the objects.

2 Related Work

The 3D printing technology provides a convenient way for fabricating objects with complex geometries, and thus draws a lot of attention in the computer graphics community. One line of this research optimizes different sub-steps of the 3D printing process, such as printing direction [Gao et al. 2015; Zhang et al. 2015b], slicing [Wang et al. 2015], decomposition and packing [Chen et al. 2015; Yao et al. 2015; Luo et al. 2012]. The other line of this research detects and fixes structural problems of the 3D printed model itself, such as stress analysis [Stava et al. 2012; Zhou et al. 2013; Umetani and Schmidt 2013] and cost-effective printing [Wang et al. 2013; Lu et al. 2014; Zhang et al. 2015a].

Generating economic and practical supporting structures is a hot topic in 3D printing technologies. Vanek et al. [2014] proposed a method to generate tree-like supporting structure to reduce material costs. Dumas et al. [2014] used scaffolding-like structures to improve the printability of overhanging parts. Hu et al. [2014] designed a method to largely reduce the necessary supporting structures by cutting the whole object into pyramidal parts. Recently, Reiner and Lefebvre [2016] proposed an interactive sculpting system for designing support-free models. All the above methods optimize supporting structures outside the object, which are relatively easy to remove after printing. In contrast, our method focuses on

(a) Support-free voxel (b) Voxelization (c) Optimization + Dithering (d) Result (e) Interior voids

Figure 2: ***Support-free voxelization.*** *(a) The support-free voxel. (b) The initial voxelization of the* ELEPHANT *using the support-free voxel as basic unit. (c) The sectional view of interior voids generated by wall thicknesses optimization and ghost voxels dithering. (d) The printing result without using support materials inside it. (e) Highlight of the interior voids. The blue point is the ground-contact point, and the transparent / solid arrows are principal axes of the input / result.*

the elimination of supporting structures inside the object, which are impossible to remove without breaking the object into pieces.

Our work is related to the interior carving methods for 3D printable objects. Prévost et al. [2013] generated statically balanced objects by voxelizing and hollowing the input model in a heuristic manner. Bächer et al. [2014] designed spinnable models by using an adaptive octree for volume voxlization and solving an optimization problem for the hollowing state of each voxel. Both above methods deform the outer surface of the object while interior hollowing itself cannot satisfy the functional requirements. Different from the volume-based methods, Musialski et al. [2015] optimized an offset surface of the input mesh as the interior shape and presented a subspace method to accelerate the computation. Inspired by previous work, we adopt an octree to voxelize the input model and design an iterative algorithm to optimize and merge voxels. However, instead of the simple cube, we design a specific support-free structure and use it as the voxel, such that the optimization result can be directly printed. Moreover, rather than optimizing the hollowing variables and then rounding them to the binary states, we use the wall thickness of each voxel as the optimization variable, which is inherently continuous.

3 Support-free Interior Carving

We introduce a computational framework for carving support-free interior voids inside an object. Taking a 3D triangular mesh as input, we first voxelize its volume by building and refining an octree. Each voxel is a support-free rhombohedron, whose central hollow part is controlled by the wall thickness (see Figure 2a, 2b). We parameterize common integral terms over each voxel as analytic functions of its wall thickness. Based on the parameterization, we formulate a continuous optimization problem for desired application by using wall thicknesses of all voxels as variables. After the optimization, we dither non-printable voxels while ensuring the whole structure is still support-free. Finally, by alternating the wall thickness optimization and voxels dithering, we obtain a hollowed model which both satisfies the desired functional purposes and can be printed out without using addtional supports inside the interior voids (see Figure 2c, 2d).

3.1 Support-free Voxel

As the basic unit of our framework, we need a voxel that: (1) is self-supportable during printing; (2) can be tiled to occupy the interior voids; (3) can change its hollowing state by computational methods.

A key observation is that a hollowed rhombohedron, i.e. a cube scaled in the diagonal direction (Figure 2a), satisfies all above requirements. In this unit structure, all the walls have the same slope angle α, which depends on the scaling factor s:

$$\alpha = \arctan(\frac{1}{2s}\sqrt{2}).$$

With sufficiently small α (we fix $\alpha = 30^\circ$), the structure is safe to print without supporting structures. Using such basic unit, we refine an octree to generate tiled voxels fully occupying the body (Figure 2b). The cavity volume of each voxel $\mathcal{V}_i$ can be adjusted by its wall thickness t_i. The voxel $\mathcal{V}_i$ is empty if $t_i = 0$ and solid if $t_i = 0.5\,l$, where l is the side length (Figure 2a). Using the wall thicknesses of the voxels, we can control the mass distribution of the model to satisfy various design targets.

3.2 Voxel Integral Parametrization

The energy functions in mass distribution optimization problems, such as make-it-stand [Prévost et al. 2013] and spin-it [Bächer et al. 2014], are composed of integral terms over the shape. For example, the center of mass $\boldsymbol{c}$ is defined as

$$\boldsymbol{c} = \frac{\int_{V-V_h} \boldsymbol{p}\, dV}{\int_{V-V_h} dV}, \tag{1}$$

where $\boldsymbol{p}$ is the position vector, V is the body of input model and V_h is the hollow parts inside it.

To formulate an analytic expression of voxel wall thicknesses, we rewrite the hollow integral in Equation (1) as a sum of integrals over each voxel:

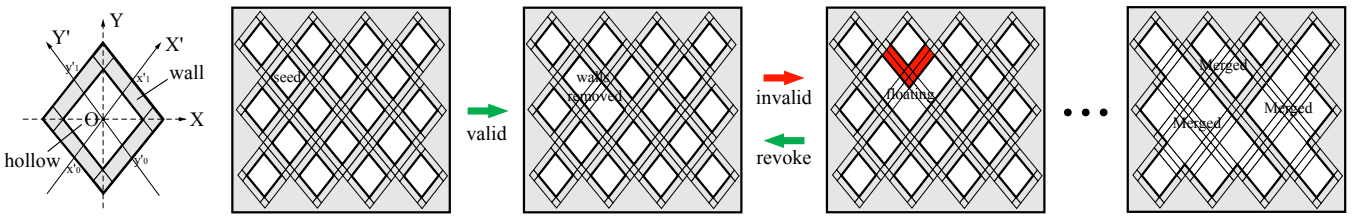
$$\int_{V_h} \boldsymbol{p}\, dV = \sum_i \int_{V_i(t_i)} \boldsymbol{p}\, dV \tag{2}$$

where t_i is the wall thickness of voxel $\mathcal{V}_i$, and $V_i(t_i)$ is the central hollow part of $\mathcal{V}_i$.

For the sake of simplicity, we use a 2D voxel (Figure 3a) to demonstrate our derivation. We perform the above integration in a local coordinate system $\mathrm{X'OY'}$ that is parallel to the voxel borders:

$$\begin{aligned}\int_{V_i(t_i)} \boldsymbol{p}\, dV &= \int_{x'_0+t_i}^{x'_1-t_i} \int_{y'_0+t_i}^{y'_1-t_i} (\vec{\boldsymbol{X}}'x' + \vec{\boldsymbol{Y}}'y')|\boldsymbol{J}|\, dy'dx' \\ &= \mathsf{c}_3 t_i^3 + \mathsf{c}_2 t_i^2 + \mathsf{c}_1 t_i + \mathsf{c}_0\end{aligned} \tag{3}$$

where $\vec{\boldsymbol{X}}'$ and $\vec{\boldsymbol{Y}}'$ are normalized vectors in direction of axes $\mathrm{X'}$ and $\mathrm{Y'}$, $\boldsymbol{J}$ is the Jacobian matrix of coordinate transformation, and x'_0, x'_1, y'_0, y'_1 are coordinates in $\mathrm{X'OY'}$ defining the integration domain of $\mathcal{V}_i$. The integration result is a cubic polynomial of variable



(a) Support-free voxel (b) Ghost voxels (c) Valid case (d) Invalid case (e) Dithering result

Figure 3: ***2D support-free voxel and dithering.*** *(a) The support-free voxel in 2D case. (b) The ghost voxels whose wall thicknesses are close to* 0*. The dithering starts from the voxel marked as "seed". (c) The walls between the seed voxel and one of its neighbour are removed. (d) An invalid removing operation is detected, as the walls highlighted in red become floating (non-printable). (e) The interior voids after dithering. The voxels are merged into three disjoint sets, while the whole structure is still support-free.*

t_i, and the coefficients c_0, c_1, c_2, c_3 are constant vectors computed from $|\boldsymbol{J}|$, $\boldsymbol{X}_0'$, $\boldsymbol{Y}_0'$, x_0', x_1', y_0', y_1'.

The above derivation can be simply generalized to different applications. In fact, we can define the integral over $V_i(t_i)$ as a functional

$$P_i(f) = \int_{V_i(t_i)} f(\boldsymbol{p})\, dV, \tag{4}$$

where the integrand $f(\boldsymbol{p})$ is application-specific. For example, the integrands used in Spin-It [Bächer et al. 2014] are listed in Equation (9). Note that no matter which integrand is used, the integration result of Equation (4) is always an analytic function of t_i. The 3D voxel parametrization is similar to the 2D case (see Appendix A for the details).

Optimization. Based on the above parameterization, we define a general optimization problem by using the wall thicknesses $\{t_i\}$ as the variables:

$$\begin{aligned} \min_{\{\boldsymbol{t_i}\}} \quad & E\left(\sum_i^m P_i(f_0), \sum_i^m P_i(f_1), \ldots, \sum_i^m P_i(f_n)\right) \\ s.t. \quad & 0 \le t_i \le 0.5\, l, \end{aligned} \tag{5}$$

where m is the number of voxels and n is the number of customizable integrands, and the energy function E defines the relationship between these integral terms for specific application.

3.3 Ghost Voxels Dithering

When the continuous optimization ends, we obtain a set of voxels with ideal wall thicknesses $t_i \in [0, 0.5\, l]$. However, due to the 3D printing precision, there exists a minimum printable thickness t_{min}. A voxel cannot be safely printed if its wall thickness is less than t_{min}.

We indicate the non-printable voxels in the optimization result as $\mathcal{G} = \{\mathcal{V}_i \,|\, t_i \le t_{min}\}$, i.e. the *ghost* set. To ensure the printability, we need to re-assign the wall thicknesses of these ghost voxels to either 0 (removed) or t_{min} (printable). We hope this change would keep the new wall thicknesses as close to the continuous optimization values as possible. Meanwhile, we must also ensure it does not lead to any floating walls (Figure 3d) that are not support-free. Inherently, this is a discrete optimization problem which has an exponential space.

We adopt a greedy strategy to tackle with this dithering (continuous to discrete) problem. Specifically, we execute a depth-first search (DFS) to gradually remove the walls between adjacent ghost voxels with thicknesses close to 0, while perserving the support-free property of the whole structure. The complete algorithm is summarized in Algorithm 1.

We use 2D case to illustrate this process (Figure 3). Starting from a ghost voxel $\mathcal{V}_i$ that has thickness close to 0 as the seed, we add it into the access record $\mathcal{A}$ and check its neighbors. For each ghost neighbor $\mathcal{V}_j$ that also has thickness close to 0, we try to remove the walls between $\mathcal{V}_j$ and its neighbors already in $\mathcal{A}$ (Figure 3c), and check if any walls on top of $\mathcal{V}_j$ become floating. If any such walls exist (Figure 3d), we add the walls removed just now back; otherwise, we add $\mathcal{V}_j$ into $\mathcal{A}$ and recursively consider its neighbors. The DFS stops when no more voxels can be added into $\mathcal{A}$ (Figure 3e). At last, we assign t_{min} to all the remaining walls of ghost voxels.

The situation in 3D is similar except that each voxel has six neighbors. Figure 2c shows a result of ghost voxels dithering on a 3D model, which is printed without using any support materials inside the interior voids.

Algorithm 1 Ghost voxels dithering.

```
procedure DFSDither(voxel V_i, set A)
    Add V_i into A
    for all ghost voxel V_j adjacent to V_i do
        if V_j ∉ A and t_j < 0.5 t_min then
            Remove walls between V_j and its neighbors in A
            if No walls on top of V_j become floating then
                DFSDither(V_j, A)
            else
                Add the removed walls back
            end if
        end if
    end for
end procedure
procedure DitherGhostVoxels(set G)
    A ← ∅                                   ▷ access record
    for all ghost voxel V_i ∈ G do
        if V_i ∉ A and t_i < 0.5 t_min then
            DFSDither(V_i, A)
        end if
    end for
    All remaining walls in G ← t_min
end procedure
```

3.4 Iterative Optimization

Our method solves an application-specific optimization problem for the wall thickness of each voxel, and then dither all the ghost voxels to ensure the printability. Most walls between ghost voxels are removed during the dithering process, only a few necessary ones are left to preserve support-free property of the whole structure. As the dithering change the wall thicknesses of ghost voxels, we adopt an iterative optimization strategy to alternate between the continuous optimization and discrete dithering, until the voxel structure does not change anymore. We list all the steps below.

Initialization. At the beginning, we voxelize the input model and precompute the parameterization coefficients of each voxel.

Continuous optimization. At the first stage of each iteration, we solve the continuous optimization (nonlinear programming) for wall thicknesses, using the range $[0, 0.5\,l]$ for box constraints.

Ghost voxels dithering. At the second stage of each iteration, we dither all the ghost voxels to convert their optimized wall thicknesses to discrete choices $\{0, t_{min}\}$. After that, for each f in the energy function we add a compensation term $\int_{V_{\Delta\text{walls}}} f(\boldsymbol{p})\,dV$ onto $\sum_i P_i(f)$, and restart the continuous optimization again. The $V_{\Delta\text{walls}}$ indicates the changed volume due to the wall differences in ghost voxels before and after dithering.

Note that we record the connected cavities (see Figure 3e) formed in previous iterations. We do not dither the ghost voxels in a connected cavity unless any such voxel has an optimized thickness $t_i > 0.5\,t_{min}$ in current iteration. We found that the strategy accelerates the dithering speed and convergence very often.

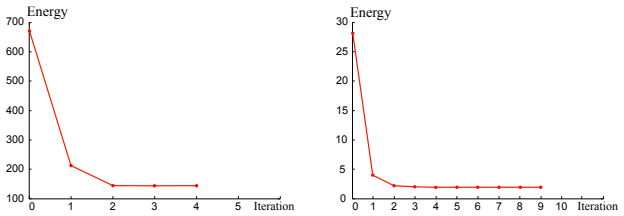
Convergence. We break the loop when the dithering result has no change between two consecutive iterations, or the maximum number of iterations is reached. Finally, we mesh the interior voids from the voxel structure for printing.

4 Applications and Results

We apply our support-free interior carving framework described in Section 3 to two design problems and print real objects for validation.

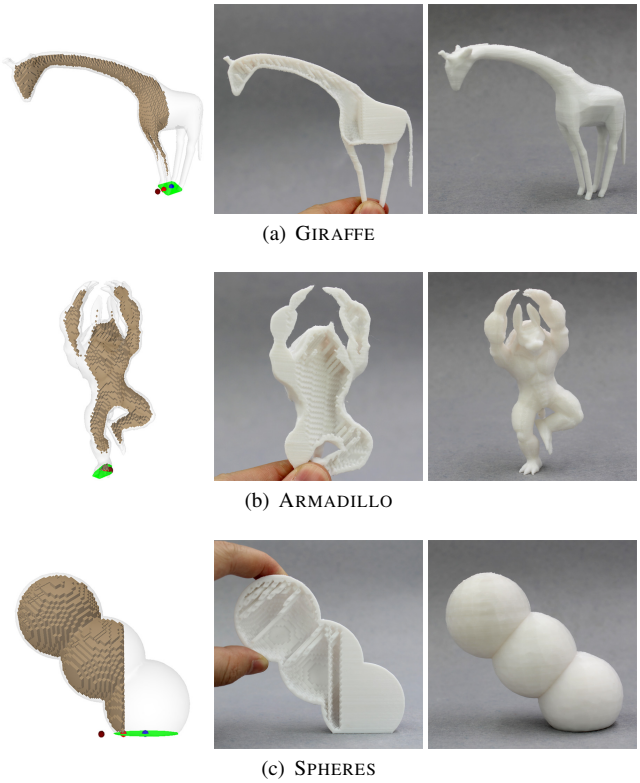
We implement and test our method on a desktop PC with Intel i7-4770 3.4GHz CPU and 16GB RAM. All programs are executed in a single thread. The BLEIC routine of the ALGLIB [Bochkanov 2013] is used to solve the continuous optimization. All results are printed by a low-cost FDM printer using PLA materials. We configure the printing software to forbid the generation of supporting structures inside the interior voids.

The object sizes are between $10cm$ and $20cm$. To ensure the shells are thick enough to print, we shrink the outer surface inward by $0.5mm$ to obtain the voxelization boundary. The minimal printable wall thickness t_{min} is set to $0.25mm$.



(a) SPHERES (b) ELEPHANT

Figure 4: *Convergence curves of our method.*



(a) GIRAFFE

(b) ARMADILLO

(c) SPHERES

Figure 5: ***Results of the static balancing application.*** *The left column highlights the computed interior voids (brown), the ground-contact region (green), the center of mass of the input (crimson), the center of mass of the result (red), and the center of ground-contact region (blue).*

4.1 Statically Balanced Object Design

Prévost et al. [2013] proposed a method for designing the static stability, which hollows an object in a heuristic way to adjust its center of mass. Due to the existence of interior voids, they have to print multiple decomposed parts to remove the supporting materials therein, and then glue them all together.

We use our optimization method to generate standing objects which can be printed as a whole. We define the objective function as the squared horizontal distance between the center of mass $\boldsymbol{c}$ and the center of ground-contact region $\boldsymbol{g}$:

$$E(\boldsymbol{t}) = (c_x(\boldsymbol{t}) - g_x)^2 + (c_y(\boldsymbol{t}) - g_y)^2, \tag{6}$$

where $\boldsymbol{t}$ is the vector stacking all voxel wall thicknesses, and $\boldsymbol{c}$ can be written as the parametrized formulation

$$\boldsymbol{c}(\boldsymbol{t}) = \frac{\int_V \boldsymbol{p}\,dV - \sum_i P_i(\boldsymbol{p})}{\int_V dV - \sum_i P_i(1)}. \tag{7}$$

Thus the objective function is an analytic function of $\boldsymbol{t}$, whose Jacobian is straightforward to derive.

Figure 1 and Figure 5 show the optimization and printing results. Note that how the centers of mass are moved from the outside to the inside of the ground-contact region by the optimization. More importantly, the generated interior voids are support-free. We can see that most walls between ghost voxels are removed to form large connected spaces, while the remaining walls form sparse pillars

Model	Tree Levels / #DOF	Voxel Size (mm)	Pre Time (s)	Poly Coef (s)	#Iter / Total Time (s)	NLP / Dithering (s)
HORSE	7 / 5,203	1.40	6.90	0.17	4 / 4.49	4.19 / 0.02
ARMADILLO	7 / 16,451	1.17	8.96	0.55	4 / 47.79	46.95 / 0.05
GIRAFFE	8 / 25,517	0.84	8.30	0.83	5 / 110.14	108.77 / 0.10
SPHERES	7 / 26,158	1.77	7.80	0.86	4 / 150.10	148.55 / 0.07
ELLIPSOID	6 / 6,278	2.29	3.57	0.25	8 / 25.67	24.94 / 0.04
ELEPHANT	7 / 10,356	1.31	8.71	0.42	9 / 42.16	41.08 / 0.07
TEAPOT	7 / 40,370	1.14	11.28	1.59	4 / 224.09	221.47 / 0.12

Table 1: ***Statistics of voxelization and performance.*** *From left to right: the number of octree levels / voxels, the voxel size, the preprocessing time, the parametrization time, the total number of iterations / the total computation time, the total time of continuous optimization / ghost voxels dithering. Level one of the octree is the bounding box of input mesh. Preprocessing includes shrinking the input mesh to get the interior boundary, building the octree, and finding the ground-contacts.*

Model	Input	Idealized	Naive	Dithered
HORSE	3.49	0.01	1.53	0.46
ARMADILLO	3.33	0.74	2.31	1.45
GIRAFFE	9.90	3.79	8.25	6.48
SPHERES	24.66	11.78	18.94	12.32

Table 2: ***Efficacy of dithering.*** *From left to right, the values are the distances (unit: mm) from target point to the center of mass of respectively the input, the continuous optimization result, the naive assignment result, and the dithering result.*

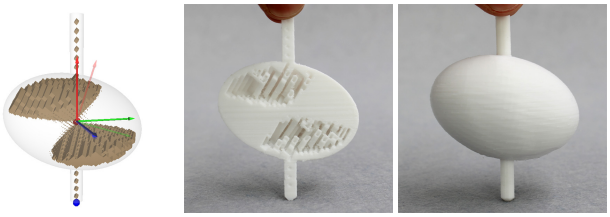


Figure 6: **ELLIPSOID.** *The left figure highlights the computed interior voids (brown), the ground-contact point (blue), the principal axes of the input (transparent arrows), and the principal axes of the result (solid arrows).*

running through the voids. The iterative optimization is crucial to the result, e.g. the interior voids of the Spheres are mainly split into the left and right parts formed in different iterations (see Figure 4 for the descent of energy).

The statistics of voxelization and performance are listed in Table 1. For all the models (top rows), our method converges within 5 iterations, while the running time depends on the number of DOFs. Compared with the continuous optimization, the parametrization and dithering take much less time.

In Table 2, we compute the horizontal distances between target point $\boldsymbol{g}$ and the centroids of four different models: (1) the input model (solid); (2) the continuous optimization result (idealized but non-printable); (3) naively assigning t_{min} to all the ghost voxel walls; (4) our dithering result. Compared with the input, the continuous optimization largely decreases the distance, which is theoretically the best result our support-free voxels can achieve. Note that we still need to trade off the distance for printability by tackeling with the ghost voxels, and apparently, our dithering algorithm keeps the centroid much closer to the target point than the naive assignment, which proves its efficacy.

4.2 Spinnable Object Design

Bächer et al. [2014] proposed a method to design the rotational stability, which uses an adaptive octree to voxelize the input model and then optimizes the fill variable of each voxel to adjust the moments of inertia. Again, a cumbersome cut-and-glue process is necessary for fabricating the design result.

We use our optimization method to generate spinnable objects that can be printed as a whole. According to Spin-It [Bächer et al. 2014], the rotational dynamics properties of an object is derived from its inertia tensor:

$$\boldsymbol{I} = \begin{pmatrix} s_{y^2} + s_{z^2} - \frac{{s_z}^2}{s_1} & -s_{xy} & -s_{xz} \\ -s_{xy} & s_{x^2} + s_{z^2} - \frac{{s_z}^2}{s_1} & -s_{yz} \\ -s_{xz} & -s_{yz} & s_{x^2} + s_{y^2} \end{pmatrix}, \quad (8)$$

where s_f are volume integrals that can be represented by our integral parametrization:

$$s_f(\boldsymbol{t}) = \int_V f \, dV - \sum_i P_i(f), \quad \text{where } f = \{1, x, y, z, x^2, y^2, z^2, xy, yz, zx\}. \quad (9)$$

Therefore, the energy (see Appendix B) derived from elements of $\boldsymbol{I}$ is a closed-form expression of $\boldsymbol{t}$.

Figure 2 and Figure 6 show our optimization results on the ELEPHANT and ELLIPSOID. Initially, the maximal principal axis (transparent arrows in Figure 2e and Figure 6 left) is not aligned with the desired rotating axis (vertical), and the center of mass is also not vertically aligned with the ground contact point. After minimizing the spinning energy, both the maximal principal axis (solid arrows) and the center of mass (solid points) satisfy the design requirements. All the fabricated objects spin well on the table (see the video). Note that there are many tiny voids in the proximity of ELLIPSOID center (Figure 6 left). This demonstrates the advantage of the continuous wall thicknesses. Since the cavity volume of each voxel vary smoothly, higher accuracy can be achieved.

The statistics of voxelization and performance for these models are reported in Table 1 (bottom rows). Similar to the design of statically balanced objects, most of the computation time is spent on the continuous optimization.

We compare our method with the state of the art interior carving method [Bächer et al. 2014] on the TEAPOT. By setting the maximal refinement level of the adaptive octree to 9 and the starting refinement level to 5, the Spin-It method converges in about 2 minutes. Figure 7 shows the generated interior voids for both methods. The two have similar overall shapes, while our result contains a few pillars to ensure the support-free property. As reported in Table 3,

Figure 7: ***Comparison with Spin-It on*** **TEAPOT.** *(left) The result of Spin-It. (right) The result of our method.*

	E_{spin}	COM	MPA
Spin-It	0.876	(0.014, 22.720, 0.028)	(-0.008, 0.999, -0.005)
Ours	1.004	(0.002, 23.192, -0.003)	(-0.002, 0.999, 0.007)

Table 3: ***Comparison on*** **TEAPOT.** *From left to right: the spinning energy, the distance from the optimized center of mass to the target point, the direction of the maximal principal axis.*

the pillars make our spinning energy a little higher than Spin-It. But note that the center of mass and the maximal principle axes are quite similar for both methods.

5 Conclusion and Discussion

We propose a novel method to carve interior voids inside objects for functional design optimization. Such objects can be directly fabricated by an FDM printer without using any support materials inside the interior voids, excluding the necessity of the tedious cut-and-glue process. We design a support-free voxel as the basic unit for voxelization and parameterize the integral terms over each voxel as closed-form expressions of its wall thickness. The wall thicknesses are then used as continuous variables for the application-specific optimization. An algorithm is developed to dither ghost voxels, ensuring printability and preserving support-free property of the object. We iteratively carve the interior voids by alternating the continuous optimization and the discrete dithering. We apply our method to design statically balanced objects and spinnable objects, and print real objects for validation.

We adopt a greedy strategy for dithering, which may get stuck in a local minimum. For such combinatorial optimization, a probabilistic technique like the simulated annealing can get better results, but at the cost of much more computation time. Our method is apt to generate more pillars when the ceilings of the interior voids are flat (Figure 7), which is probably less optimized. Our result also depends on the printing orientation, thus finding a best orientation for the support-free structure can be an interesting future work.

References

BÄCHER, M., WHITING, E., BICKEL, B., AND SORKINE-HORNUNG, O. 2014. Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph. 33*, 4, 96.

BOCHKANOV, S., 2013. Alglib (www.alglib.net).

CHEN, X., ZHANG, H., LIN, J., HU, R., LU, L., HUANG, Q., BENES, B., COHEN-OR, D., AND CHEN, B. 2015. Dapper: decompose-and-pack for 3d printing. *ACM Trans. Graph. 34*, 6, 213.

CHRISTIANSEN, A. N., SCHMIDT, R., AND BÆRENTZEN, J. A. 2015. Automatic balancing of 3d models. *Computer-Aided Design 58*, 236–241.

DUMAS, J., HERGEL, J., AND LEFEBVRE, S. 2014. Bridging the gap: Automated steady scaffoldings for 3d printing. *ACM Trans. Graph. 33*, 4, 98.

GAO, W., ZHANG, Y., NAZZETTA, D. C., RAMANI, K., AND CIPRA, R. J. 2015. Revomaker: Enabling multi-directional and functionally-embedded 3d printing using a rotational cuboidal platform. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ACM, 437–446.

HU, R., LI, H., ZHANG, H., AND COHEN-OR, D. 2014. Approximate pyramidal shape decomposition. *ACM Trans. Graph. 33*, 6, 213.

LU, L., SHARF, A., ZHAO, H., WEI, Y., FAN, Q., CHEN, X., SAVOYE, Y., TU, C., COHEN-OR, D., AND CHEN, B. 2014. Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph. 33*, 4, 97.

LUO, L., BARAN, I., RUSINKIEWICZ, S., AND MATUSIK, W. 2012. Chopper: Partitioning models into 3D-printable parts. *ACM Trans. Graph. 31*, 6 (Dec.).

MUSIALSKI, P., AUZINGER, T., BIRSAK, M., WIMMER, M., AND KOBBELT, L. 2015. Reduced-order shape optimization using offset surfaces. *ACM Trans. Graph. 34*, 4, 102.

PRÉVOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make it stand: balancing shapes for 3d fabrication. *ACM Trans. Graph. 32*, 4, 81.

REINER, T., AND LEFEBVRE, S. 2016. Interactive modeling of support-free shapes for fabrication. In *Eurographics (short papers)*, Eurographics Association.

STAVA, O., VANEK, J., BENES, B., CARR, N., AND MĚCH, R. 2012. Stress relief: improving structural strength of 3d printable objects. *ACM Trans. Graph. 31*, 4, 48.

UMETANI, N., AND SCHMIDT, R. 2013. Cross-sectional structural analysis for 3d printing optimization. *SIGGRAPH Asia 5*, 1–4.

VANEK, J., GALICIA, J. A., AND BENES, B. 2014. Clever support: Efficient support structure generation for digital fabrication. *Computer graphics forum 33*, 5, 117–125.

WANG, L., AND WHITING, E. 2016. Buoyancy optimization for computational fabrication. *Computer Graphics Forum 35*, 2.

WANG, W., WANG, T. Y., YANG, Z., LIU, L., TONG, X., TONG, W., DENG, J., CHEN, F., AND LIU, X. 2013. Cost-effective printing of 3d objects with skin-frame structures. *ACM Trans. Graph. 32*, 6, 177.

WANG, W., CHAO, H., TONG, J., YANG, Z., TONG, X., LI, H., LIU, X., AND LIU, L. 2015. Saliency-preserving slicing optimization for effective 3d printing. *Computer Graphics Forum 34*, 6, 148–160.

WANG, W., ZANNI, C., AND KOBBELT, L. 2016. Improved surface quality in 3d printing by optimizing the printing direction. *Computer Graphics Forum 35*, 2.

YAO, M., CHEN, Z., LUO, L., WANG, R., AND WANG, H. 2015. Level-set-based partitioning and packing optimization of a printable model. *ACM Trans. Graph. 34*, 6, 214.

ZHANG, X., XIA, Y., WANG, J., YANG, Z., TU, C., AND WANG, W. 2015. Medial axis tree–an internal supporting structure for 3d printing. *Computer Aided Geometric Design 35*, 149–162.

ZHANG, X., LE, X., PANOTOPOULOU, A., WHITING, E., AND WANG, C. C. 2015. Perceptual models of preference in 3d printing direction. *ACM Trans. Graph. 34*, 6, 215.

ZHOU, Q., PANETTA, J., AND ZORIN, D. 2013. Worst-case structural analysis. *ACM Trans. Graph. 32*, 4, 137.

A Coefficients of 3D Voxels Parameterization

Here we describe how to compute all the coefficients for 3D voxels parametrization. The general form of the integrals over a voxel $\mathcal{V}_i$ is

$$P_i(f) = \int_{x_0+t_i}^{x_1-t_i} \int_{y_0+t_i}^{y_1-t_i} \int_{z_0+t_i}^{z_1-t_i} f(\boldsymbol{p})\, dzdydx,$$

where $f(\boldsymbol{p}) = \{1, x, y, z, x^2, y^2, z^2, xy, yz, zx\}$, and x, y, z are coordinates of the 3D position vector $\boldsymbol{p}$. By integrating them in the boundary-aligned coordinate system of $\mathcal{V}_i$ (similar to Figure 3a), we get polynomials of t_i.

We do not need to derive all the closed-form formulas manually. By substituting the above $f(\boldsymbol{p})$ into the integral, one could find that there are four general forms:

$$T(t_i) = \begin{cases} \int_{x_0+t_i}^{x_1-t_i} \int_{y_0+t_i}^{y_1-t_i} \int_{z_0+t_i}^{z_1-t_i} |\boldsymbol{J}|\, dz'dy'dx' \\ \int_{x_0+t_i}^{x_1-t_i} \int_{y_0+t_i}^{y_1-t_i} \int_{z_0+t_i}^{z_1-t_i} A_m\alpha|\boldsymbol{J}|\, dz'dy'dx' \\ \int_{x_0+t_i}^{x_1-t_i} \int_{y_0+t_i}^{y_1-t_i} \int_{z_0+t_i}^{z_1-t_i} A_mB_m\alpha^2|\boldsymbol{J}|\, dz'dy'dx' \\ \int_{x_0+t_i}^{x_1-t_i} \int_{y_0+t_i}^{y_1-t_i} \int_{z_0+t_i}^{z_1-t_i} A_mB_n\alpha\beta|\boldsymbol{J}|\, dz'dy'dx' \end{cases},$$

where

$$A, B = \{X', Y', Z'\}, \quad m, n = \{0, 1, 2\}, \quad \alpha, \beta = \{x, y, z\}.$$

We write programs to compute the above terms, which take different combinations of $A, B, m, n, \alpha, \beta$ as inputs and output the coefficients of the integration results (fifth-order polynomials).

B Energy of Spinnable Object Design

The spinning energy term E_{spin} is formulated as:

$$E_{spin} = (\frac{I_a}{I_c})^2 + (\frac{I_b}{I_c})^2,$$

where I_a, I_b, I_c are the principal moments of inertia and $I_c = I_{33}$. The I_a and I_b can be computed as:

$$I_a, I_b = \frac{1}{2}(I_{11} + I_{22} \pm \sqrt{(I_{11} - I_{22})^2 + 4I_{12}^2}),$$

where I_{ij} is the ij entry of tensor $\boldsymbol{I}$. To make an object spinnable, we still need to align its center of mass with the ground contact point vertically and align its maximal principal axis to the rotating axis (z axis). We achieve this by adding two penalty terms

$$E_{com} = (c_x)^2 + (c_y)^2, \quad E_{axis} = (\frac{I_{13}}{I_{33}})^2 + (\frac{I_{23}}{I_{33}})^2$$

into the objective function, such that the optimization is formulated as:

$$\begin{aligned} \min \quad & \omega_{spin}E_{spin} + \omega_{com}E_{com} + \omega_{axis}E_{axis} \\ s.t. \quad & 0 \le t_i \le 0.5\, l, \end{aligned}$$

where ω_{spin}, ω_{com} and ω_{axis} are the relative weights.