Front. Comput. Sci., 2017, 11(2): 332–346
DOI 10.1007/s11704-016-5465-y

RESEARCH ARTICLE

Continuous optimization of interior carving in 3D fabrication

Yue XIE[1], Ye YUAN[1], Xiang CHEN (✉)[1], Changxi ZHENG[2], Kun ZHOU[1]

1 State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China
2 Columbia University, New York NY 10027, USA

Abstract In this paper we propose an optimization framework for interior carving of 3D fabricated shapes. Interior carving is an important technique widely used in industrial and artistic designs to achieve functional purposes by hollowing interior shapes in objects. We formulate such functional purpose as the objective function of an optimization problem whose solution indicates the optimal interior shape. In contrast to previous volumetric methods, we directly represent the boundary of the interior shape as a triangular mesh. We use Eulerian semiderivative to relate the time derivative of the object function to a virtual velocity field and iteratively evolve the interior shape guided by the velocity field with surface tracking. In each iteration, we compute the velocity field guaranteeing the decrease of objective function by solving a linear programming problem. We demonstrate this general framework in a novel application of designing objects floating in fluid and two previously investigated applications, and print various optimized objects to verify its effectiveness.

Keywords computer graphics, 3D printing, interior carving, shape optimization, Eulerian semiderivative

1 Introduction

Interior shape carving has been proved as an effective method for achieving various design purposes without affecting the exterior appearances of objects. It has widespread use in mechanical and industrial designs for satisfying functional requirements, decreasing product weights and reducing manufacturing costs. Despite its effectiveness, hollowing an object, even for a simple goal like adjusting the center of mass, requires a trial-and-error process, which is often time consuming. For example, it is hard, if not impossible, to control the position of the center of mass by manually adjusting the hollowed shape. Recent research [1,2] has seen computational methods proposed to overcome such difficulties. Through voxelization-based representation and optimization, such methods are successfully applied to design statically standing objects and dynamically spinning tops.

While voxelization is an effective spatial discretization for shapes, this representation has essentially a binary nature: an voxel is either fully occupied by printing material or not. As a result, the boundary of a voxel grid is always rugged no matter how fine the voxelization is, and a fine-granular voxel grid is always needed to achieve high approximation accuracy. On the other hand, the interior carving is the common requirements of various design problems. To minimize the application-specific customization efforts, it is necessary to keep a unique interior carving method for different computational design tasks.

In this paper, we propose a general optimization framework for interior carving that represents the interior boundary as an explicit triangular mesh. Unlike the volumetric discretization, either voxel grid or tetrahedral mesh, adopted in previous methods [1–3], our optimization framework involves only the numerical discretization on the carving boundary using triangular meshes. This is advantageous to the accuracy of carving boundary. We show that our method often obtains lower optimization energies compared with the spin-it method, for various application scenarios (see comparisons in Section 5). Our method not only reduces the de-

Received November 4, 2015; accepted March 28, 2016

E-mail: xchen.cs@gmail.com

grees of freedom necessary for shape representation and optimization, but also enables the direct controlling of surface properties of the carving boundary, which is not accessible with volumetric discretization. Figure 1 (b) shows an example of smoothness control for the interior surface. Based on the surface discretization, we present a novel continuous optimization approach. We formulate the design purposes as objective functions based on surface integrals, and then iteratively advance the boundary mesh to minimize the objective functions. Specifically, we develop an algorithm based on Eulerian semiderivative to compute an optimal velocity field in which the triangular mesh is deformed. Such deformation ensures that the objective function is decreased fastest, i.e., in a gradient descend manner. Using this optimization strategy, the boundary of the interior voids is continuously varied to satisfy the design requirements.

Our framework is versatile and flexible. A variety of design constraints for different problems can be easily incorporated. We present a new application of designing floating objects, whose usage ranges from ocean sciences to consumer products. We formulate the buoyancy related objective function and constraints and optimize the interior carving to balance an object floating on liquid surface with specific pose (Fig. 1(c)). Using an FDM 3D printer, we fabricate the optimized objects and demonstrate that our optimization method is able to generate interior carved shapes accurately satisfying the design constraints. We also apply our optimization method to the design for statically balanced and spinnable objects. Experiments on several examples are executed to show the effectiveness of the continuous optimization framework.

2 Related work

The design optimization technologies have drawn a lot of attention in the structural and mechanical engineering field. This line of research exerts and optimizes varying variations such as feature size [4], boundary geometry [5,6], structural topology [7,8] or mixed types [9], to satisfy specific engineering requirements such as product states in response to external loads. We refer to the surveys [10,11] for a comprehensive discussion of structural optimization in engineering.

With the increasing popularity of the additive manufacturing (or 3D printing) technologies, the computer graphics community shows growing research interests in designing physically fabricable shapes. Several methods are presented to decompose 3D shapes to facilitate the fabricating process [12,13], or fold 3D shapes for packing [14]. Research efforts have also been made to bring virtual characters into real world, e.g., deformable objects [15–17], articulated models [18,19] and mechanical characters [20,21]. Another subset of works focused on the physical soundness of fabricated shapes, like the plank-based furniture [22], the self-supporting surfaces [23–25], cost-effective printing [26,27], structural weakness detection and reinforcement [28–30].

Recently, optimizing mass properties draws the attention of researchers. Prévost et al. [1] adopted grid of voxels to discretize the interior volume of an object and used a heuristic optimization method to adjust the center of mass of the object. Bächer et al. [2] used an adaptive octree to refine the voxelization and solved for the binary states of the voxels as the optimal solution for adjusting the rotational dynamic properties of rigid bodies. Christiansen et al. [3] proposed a tetrahedral mesh based hollowing optimization for balancing static objects. Unlike these previous methods which adopt volumetric discretization, our interior carving method is purely based on surface discretization and integrals. We represent the boundary of the interior void as a triangular mesh, which has much less degrees of freedom to handle and is more accurate for complex geometries. In addition, surface properties such as the smoothness of the carving shape can be explicitly controlled.

As a concurrent work, Musialski et al. [31] proposed a shape optimization method which offsets the outer surface of an object to generate an interior surface as the boundary of

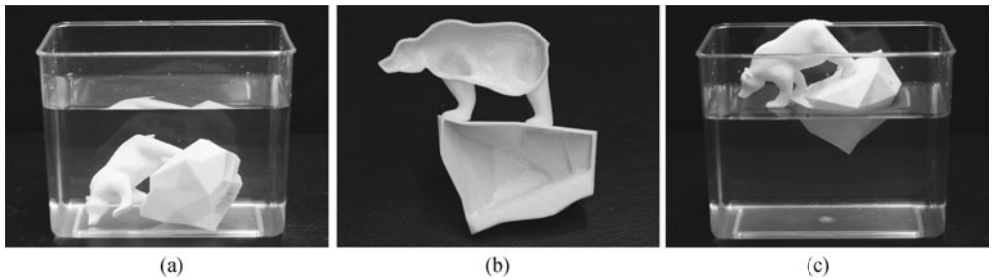


Fig. 1 An example of our floatation application. (a) Unhollowed model (before interior carving, the model sinks); (b) interior carving (we carve an interior shape generated by our algorithm inside the model); (c) hollowed model (after the interior shape is hollowed, the model floats in the water with a specific pose)

the interior voids. Their method utilizes the manifold harmonics to construct a set of low-frequency basis of the surface offset space and then solves a reduced optimization problem. Although the method increases the robustness and reduces the optimization complexity with an elaborately chosen subspace, a robust skeleton is required for computing the offset space and the final carving shape is limited to it. Instead, our continuous optimization method utilizes surface tracking to directly advance the surface of the interior voids and adaptively changes the mesh structure, such that our carving shape can be more freeform.

3 Optimization of interior carving

In this section we present the general framework of our algorithm for interior carving optimization. It is then reified with specific formulations for different applications in Section 4.

The input of our algorithm is the outer surface of a 3D model and an initial surface representing the boundary of the interior shape to carve, and both are 2D manifolds represented by triangular meshes. Our algorithm iteratively changes the geometry and possible topology of the interior shape to minimize an objective function while satisfying constraints. In each iteration, a linear programming problem is solved for a virtual velocity field, which is then utilized to advance the boundary of the interior shape. The output is an triangular mesh representing the optimized interior shape which is ready to be used in 3D fabrication. This optimization framework is general, as a large class of objectives and constrains can be incorporated in a unified way.

The entire computation is performed directly on triangular meshes, and hence no volumetric discretization is needed. The whole method is simple to implement, as outlined in Algorithm 1. In the following subsections, we describe each step of the algorithm in detail.

Algorithm 1 Gradient descent optimization for interior carving

```
Require: initial interior shape Ω
  procedure CarvingOptimization(Ω)
    ΔJ ← ∞
    while ΔJ > ε do
      compute P(x) at vertices using Eq. (13).        Section 3.2
      solve LP problem Eq. (17) for v_n.              Section 3.3
      advance Ω as Ω + v_n n Δt.                      Section 3.4
      update J and the relative change ΔJ.
    end while
  end procedure
```

3.1 Optimization problem formulation

For a hollowed solid body (Fig. 2), as in the inset, its center

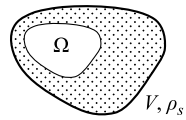


Fig. 2 A hollowed solid body

of mass $\boldsymbol{c}$ is computed as

$$\boldsymbol{c} = \frac{\int_V \rho_s \boldsymbol{x} \mathrm{d}v - \int_\Omega \rho_s \boldsymbol{x} \mathrm{d}v}{\int_V \rho_s \mathrm{d}v - \int_\Omega \rho_s \mathrm{d}v}, \tag{1}$$

where ρ_s is the density of the solid body, $\boldsymbol{x}$ is position coordinate, V is the volume of the whole input body and Ω represents the volume of the interior shape to hollow. To move $\boldsymbol{c}$ towards a constant target point $\boldsymbol{p}$, we can define a simple objective function J measuring the squared distance between $\boldsymbol{c}$ and $\boldsymbol{p}$,

$$J = |\boldsymbol{c} - \boldsymbol{p}|^2. \tag{2}$$

By optimizing the shape of the hollowed void Ω, we minimize J thus accomplish the adjustment of the center of mass.

The fundamental term in such objective functions is the volume integral (see Eq. (1)). If we denote the volume integral as

$$I_{(f;V)} = \int_V f(\boldsymbol{x}) \mathrm{d}v, \tag{3}$$

where $f(\boldsymbol{x})$ could be any scalar function defined on the volume V, then Eq. (1) can be rewritten as

$$c_\alpha = \frac{I_{(\rho_s \alpha;V)} - I_{(\rho_s \alpha;\Omega)}}{I_{(\rho_s;V)} - I_{(\rho_s;\Omega)}}, \quad \text{where } \alpha = x, y, z, \tag{4}$$

and the objective function Eq. (2) can be expressed as

$$J = J(I_{(\rho_s x;\Omega)}, I_{(\rho_s y;\Omega)}, I_{(\rho_s z;\Omega)}, I_{(\rho_s;\Omega)}), \tag{5}$$

which is a function of Ω.

A large class of objective functions can be expressed in a form similar to Eq. (5), as for the applications in Section 4. For such applications, we seek an optimized interior shape Ω for carving to minimize an objective function J. Thus the general form of our optimization problem is defined as

$$\min_\Omega J(I_1, I_2, \ldots, I_n). \tag{6}$$

The interior shape Ω should also satisfy certain constraints, such as surface thickness requirement and volume preservation. Below we first present an algorithm to minimize the objective function Eq. (6) in Section 2, and then describe how to incorporate hard constraints in Section 3.

3.2 Gradient descent using Eulerian semiderivative

We now present an approach to solve the optimization problem Eq. (6). Following the general spirit of gradient descent,

we iteratively changes the interior shape by advancing its boundary to decrease the objective function until the change of the objective function drops below a threshold. As in any gradient descent method, the fundamental question is how to find the gradient descent direction, which in our case means how to find the fastest way of advancing the boundary to decrease the objective function.

To find the gradient descent direction, we borrow a concept named Eulerian semiderivative from shape optimization [32]. Considering a volume shape Ω changes with time, the Eulerian semiderivative of a functional defined on Ω, such as our volume integral Eq. (3), is defined as

$$dI_{(f;\Omega^0)} = \lim_{t \downarrow 0} \frac{I_{(f;\Omega^t)} - I_{(f;\Omega^0)}}{t}, \tag{7}$$

where Ω^0 is the shape of Ω at time 0 and Ω^t is its shape at time t. Similar to a derivative which tells us how fast a function changes with respect to a variable, the Eulerian semiderivative measures how fast $I_{(f,\Omega)}$ changes with respect to the volume shape Ω.

Assume Ω is deformed by a velocity field

$$\boldsymbol{v}(\boldsymbol{x}) = \frac{\partial \boldsymbol{x}}{\partial t}, \quad \text{where } \boldsymbol{x} \in \Omega, \tag{8}$$

where $\boldsymbol{x}$ is the position of an arbitrary point in Ω and $\boldsymbol{v}(\boldsymbol{x})$ is the velocity of that point. It is proven in Delfour and Zolésio [32] that if the velocity field $\boldsymbol{v}$ is smooth enough, the Eulerian semiderivatives of I at time 0 can be calculated as

$$\mathrm{d}I_{(f;\Omega^0)} = \int_{\Gamma^0} f(\boldsymbol{x})\boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) \mathrm{d}\Gamma, \tag{9}$$

where Γ^0 represents the boundary of Ω^0 and $\boldsymbol{n}(\boldsymbol{x})$ is the surface normal at $\boldsymbol{x}$. Under the same smooth velocity field assumption, a composite functional J such as Eq. (6) can be calculated using the chain rule

$$\mathrm{d}J(I_1, I_2, \ldots, I_n) = \sum_{k=1}^{n} \left.\frac{\partial J}{\partial I_k}\right|_{\Omega^0} \mathrm{d}I_k. \tag{10}$$

For the carving applications, Eq. (9) and Eq. (10) provide us a way to relate the derivative of an objective function J to a velocity field $\boldsymbol{v}$ which is defined on the boundary Γ of the interior carving shape Ω. Instead of dealing with the shape of Ω directly, we decrease J by finding a $\boldsymbol{v}$ that decreases dJ and then advance Γ with such $\boldsymbol{v}$.

In Eq. (9), what really matters is the projection of $\boldsymbol{v}(\boldsymbol{x})$ on $\boldsymbol{n}(\boldsymbol{x})$. Thus instead of an arbitrary velocity field, we always advance the boundary in a velocity field normal to the interior boundary

$$\boldsymbol{v}(\boldsymbol{x}) = v_n(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x}), \tag{11}$$

where $v_n(\boldsymbol{x})$ is a scalar field representing the normal velocity magnitude, as shown in Fig. 3. This is similar to a surface normal flow in differential geometry [33,34].

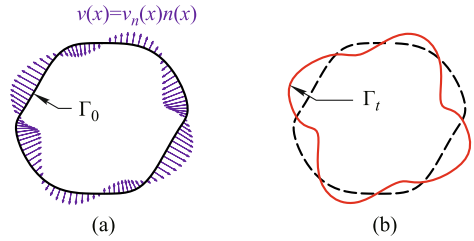


Fig. 3 Evolution of interior carving boundary in a single gradient-descent step. (a) Before an iteration step (normal velocity is defined on the boundary); (b) after an iteration step (a surface normal flow is used to evolve the carving boundary)

Putting all these together, we relate the Eulerian semiderivative of the objective function J to a scalar field v_n defined on Γ as

$$\mathrm{d}J(I_1, I_2, \ldots, I_n) = \int_{\Gamma^0} P(\boldsymbol{x}) v_n(\boldsymbol{x}) \mathrm{d}\Gamma, \tag{12}$$

where

$$P(\boldsymbol{x}) = \sum_{k=1}^{n} f_k(\boldsymbol{x}) \left.\frac{\partial J}{\partial I_k}\right|_{\Omega^0}, \tag{13}$$

and advance the interior boundary by the normal velocity field $v_n\boldsymbol{n}$ to decrease J. If there are no constraints applied to v_n, we can simply set $v_n(\boldsymbol{x})$ as $-P(\boldsymbol{x})$ which always leads to a non-positive dJ. We show in Appendix A that the $v_n(\boldsymbol{x})$ decreases J fastest, i.e., the gradient descent direction, is indeed consistent with $-P(\boldsymbol{x})$. In practice, there are always additional constraints that need to be satisfied. Therefore we solve a constrained optimization problem in each iteration to find the v_n leading to the minimal dJ (see Section 3.3).

After v_n is computed, the boundary Γ is advanced as

$$\boldsymbol{x} = \boldsymbol{x} + v_n(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x})\Delta t, \ \forall \boldsymbol{x} \in \Gamma, \tag{14}$$

where Δt is the timestep size. Such boundary advancing finishes a gradient descent step and moves the algorithm on to the next iteration.

3.3 Constrained optimization

In many applications, the optimized objects also need to satisfy certain constraints. As demonstrated in Section 4, many application specific requirements can be formulated as linear equalities and inequalities with respect to the scalar field v_n. In general, our optimization can incorporate a set of con-

straints expressed as

$$\begin{aligned}&\int_\Gamma A_i(\boldsymbol{x})v_n(\boldsymbol{x})\mathrm{d}\Gamma = C_i,\ i = 1, 2, \ldots, N_a,\\ &\int_\Gamma B_j(\boldsymbol{x})v_n(\boldsymbol{x})\mathrm{d}\Gamma \leqslant D_j,\ j = 1, 2, \ldots, N_b,\end{aligned} \tag{15}$$

where A_i and B_j are known functions, and C_i, D_j are constants. Consequently, we need to find a v_n which satisfies these constraints and meanwhile points as closely as possible toward the gradient descent direction. Concretely, we solve the optimization problem

$$\begin{aligned}&\min_{v_n} \int_{\Gamma^0} P(\boldsymbol{x})v_n(\boldsymbol{x})\mathrm{d}\Gamma \text{ as in Eq. (12)}\\ &\text{s.t. all the constraints Eq. (15),}\end{aligned} \tag{16}$$

to obtain such v_n.

Surface discretizaton In practice, the interior boundary Γ is represented by a triangle mesh and the surface integrals are discretized as weighted summations over all triangles. Therefore we discretize $P(\boldsymbol{x})$, $v_n(\boldsymbol{x})$, $A_i(\boldsymbol{x}), i = 1, 2, \ldots, N_a$ and $B_j(\boldsymbol{x}), j = 1, 2, \ldots, N_b$, by stacking their values on vertices into vectors, as $\boldsymbol{P}$, $\boldsymbol{v}_n$, $\boldsymbol{A}_i$ and $\boldsymbol{B}_j$ respectively. The solution to Eq. (16) is then approximated by solving a standard linear programming (LP) problem:

$$\begin{aligned}\min\quad & \boldsymbol{P}\cdot\boldsymbol{v}_n\\ s.t.\quad & \boldsymbol{A}_i\cdot\boldsymbol{v}_n = C_i,\ \ i = 1, 2, \ldots, N_a,\\ & \boldsymbol{B}_j\cdot\boldsymbol{v}_n \leqslant D_j,\ \ j = 1, 2, \ldots, N_b.\end{aligned} \tag{17}$$

Such a linear programming problem can be efficiently solved in polynomial time [35]. In our implementation, we adopt the widely used library GLPK [36] to solve these problems.

3.4 Interior surface advancing

In each iteration step, our algorithm advances the interior boundary Γ with the computed normal velocity field $v_n(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x})$.

Surface tracking Advancing a surface using a velocity field is a typical yet nontrivial surface tracking problem, since the topology changes such as merging and pinching-off may occur during the surface advancement. A widely used approach is the fast marching algorithm which represents the surface implicitly using level-sets [34,37]. But this method needs to extrapolate the velocity field from the surface into the entire volume, and its inherent numerical dissipation renders the equality constraints hard to satisfy precisely.

We use the explicit surface tracking approach [38] recently developed for surface tracking in fluid simulation. It advances a triangular surface using explicit forward Euler methods, and then carefully handles the collisions and topology changes.

Step length To ensure robust gradient descent and surface tracking, we set limits on how much the interior shape can change in each iteration step. Specifically, we enforce an upper limit u for the distance a vertex can advance in a single step, by using box constraints

$$-\frac{u}{\Delta t} \leqslant v_n(\boldsymbol{x}) \leqslant \frac{u}{\Delta t},\ \forall \boldsymbol{x} \in \Gamma. \tag{18}$$

As in a standard gradient descent process, if the objective function increases after the surface advancement, we roll back the surface changes and try again with a halved upper limit $(1/2)u$. When u is successively decreased to a value below a minimum step threshold u_t, the main loop terminates.

Our implementation adopts an initial upper limit $u_0 = (1/500)D$ where D is the diameter of the model. The threshold u_t is set as $(1/8)u_0$. The choice of timestep size Δt is irrelevant to the optimization result, and we simply set it as the constant 1.0.

4 Applications

In this section, we apply the general optimization algorithm developed in Section 3 to a novel application for designing floating objects and two existing applications investigated in previous works. We first introduce some common notations and constraints used in all three applications, and then formulate the specific objective functions and constraints for each application respectively.

4.1 Common settings

Notations For convenience, we use the notation s_f to represent the volume integration on the hollowed object

$$s_f = I_{(\rho_s f;V)} - I_{(\rho_s f;\Omega)}. \tag{19}$$

Then the formula of the center of mass Eq. (4) can be compactly written as

$$c_\alpha = \frac{s_\alpha}{s_1},\quad \text{where } \alpha = x, y, z. \tag{20}$$

Here s_1 means the integrand f is the constant 1, and thus s_1 equals the mass of the hollowed object. These terms are frequently used in all three applications.

The Eulerian semiderivatives of these basic terms are listed in the Appendix B. With these terms, the Eulerian semiderivatives of the objective functions defined in this section can be derived in a straightforward manner using the chain rule. The only exception is the Eq. (38) in Section 4.3,

whose Eulerian semiderivative is complex and is thus listed in Appendix B.

Coordinate frame In all applications we assume a global coordinate frame whose x/y axis lies in the horizontal plane and z axis points to the upward direction.

Thickness requirement To ensure that the results can be fabricated successfully by a 3D printer, we use inequality constraints to enforce that the wall thickness of the hollowed models is no less than a specified threshold m.

Let $\boldsymbol{x}_i$, $\boldsymbol{n}_i$ and $v_{n,i}$ denote respectively the position, normal direction, and normal velocity magnitude of a vertex v_i on the boundary triangle mesh, and $\phi(\boldsymbol{x}_i)$ denotes the distance of v_i from its position $\boldsymbol{x}_i$ to the outer surface of the object. We require that every vertex, after each forward-Euler step of surface advancing, has a distance value larger than m

$$\phi(\boldsymbol{x}_i + v_{n,i}\boldsymbol{n}_i\Delta t) \geqslant m. \tag{21}$$

For small displacements, we can linearize this constraint and obtain

$$\phi(\boldsymbol{x}_i) + \nabla\phi(\boldsymbol{x}_i)\cdot\boldsymbol{n}_i v_{n,i}\Delta t \geqslant m. \tag{22}$$

Here $\phi(\boldsymbol{x}_i)$ and $\nabla\phi(\boldsymbol{x}_i)$ can be easily computed using standard approaches such as Kd-tree and numerical methods such as central difference. Equation (22) is used in each iteration as an inequality constraint.

Initial interior shape To start the algorithm in Section 3, we construct an initial interior shape by shrinking the input outer mesh. Using the surface tracking method as in Section 3.4, we advance the outer mesh along the negative normal directions by a distance of the thickness requirement m to obtain such initial interior boundary.

Surface properties Representing the interior boundary as an explicit triangular mesh enables us to exert extra controls on the carving surface. For example, we add an energy term measuring the smoothness of the boundary surface into our objective functions, by adopting the Laplacian coordinates [39]:

$$E_L = |\boldsymbol{L}\boldsymbol{x}|^2\ ,\ L_{ij} = \begin{cases} 1, & i = j; \\ -\dfrac{1}{d_i}, & \mathrm{v}_j \in \mathcal{N}(\mathrm{v}_i); \\ 0, & \text{otherwise}, \end{cases} \tag{23}$$

where $\boldsymbol{L}$ is the Laplacian matrix, $\mathcal{N}(\mathrm{v}_i)$ is the 1-ring neighborhood of v_i, and d_i is the total number of vertices in $\mathcal{N}(\mathrm{v}_i)$. The time derivative of E_L is computed as

$$\frac{\mathrm{d}E_L}{\mathrm{d}t} = \boldsymbol{l}\cdot\boldsymbol{v}_n,\ \ \boldsymbol{l}_k = 2\sum_i\sum_j L_{ik}L_{ij}\boldsymbol{x}_j\cdot\boldsymbol{n}_k. \tag{24}$$

Equation (24) can be easily added into the linear system Eq. (16) as an additional energy term. We show how this energy term can affect boundary smoothness with different weights in Section 5.

4.2 Design of floating objects

As a novel application, we apply our general algorithm introduced in Section 3 to the design of objects floating in fluid. Since the earliest man-made boats crafted at prehistoric times [40], floating bodies have profoundly influenced human society, ranging from marine science and ocean industries to consumer products. Although lots of tools have been established to well control the floating states and stabilities of structured shapes such as ships and offshore equipments, the design of arbitrarily shaped floating objects remains to be a laborious trial-and-error process. Very recently, Musialski et al. [31] applied their subspace surface offsetting method to this application, but they assumed the objects were fully immersed in the fluid. In our work, we allow users to design objects floating above the fluid level by specifying its pose and immersion height.

The input of the application is a triangle mesh representing the outer surface and pose of a floating object, plus its immersion depth h (see Fig. 4(a)). Given the outer surface and the immersion height, the volume of the immersed part of the object is determined (the green part). The buoyancy force B and the center of buoyancy $\boldsymbol{b}$ are computed as

$$B = I_{(\rho_f;V_f)}g, \tag{25}$$

and

$$b_\alpha = \frac{I_{(\rho_f\alpha;V_f)}}{I_{(\rho_f;V_f)}},\quad \text{where } \alpha = x, y, z, \tag{26}$$

where ρ_f is the density of the fluid, g is the gravity acceleration constant and V_f is the volume of the immersed part.

The expected output is a hollowed object that floats in fluid with the desired immersion depth h and the same pose as the input mesh. We compute the interior (hollowed) shape by three successive steps. In each step we solve an individual optimization problem while preserving the results of previous steps.

Step 1 To make the object float in the fluid with the specified immersion depth h, we first balance the magnitude of the hollowed object's gravity against the magnitude of the buoyancy force. Therefore, we minimize the difference between the gravity and the buoyancy force

$$J_{volume} = \frac{1}{2}(I_{(\rho_s;V)}g - I_{(\rho_s;\Omega)}g - B)^2, \tag{27}$$

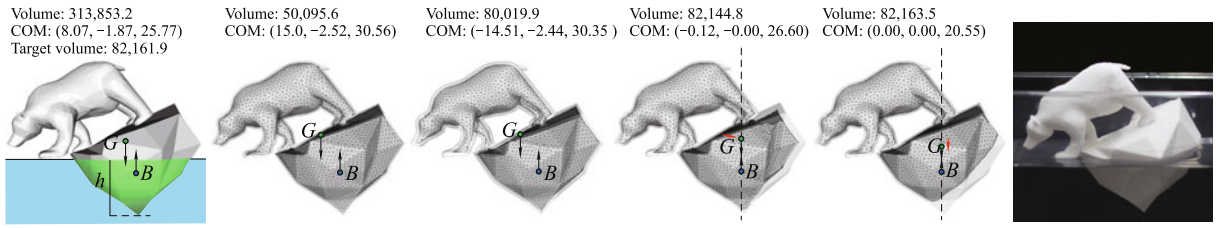


Fig. 4 The workflow of floatation application (The object is translated such that the center of buoyancy (COB) is put at the origin of the world coordinatesystem. The target volume is the model volume required to balance gravity and buoyancy). (a) Input; (b) initial interior shapes; (c) after balancing gravity and buoyancy force; (d) after aligning COM and COB; (e) after lowering down COM; (f) print

such that the thickness constraint Eq. (22) are satisfied at the same time. An example of the interior shape after this step is shown in Fig. 4(c).

Step 2 To make the object float in the same pose as the input mesh, the center of mass of the hollowed object must align vertically with the center of buoyancy. Thus we set the objective function of Step 2 as the squared distance between the two centers in the horizontal plane

$$J_{align} = \frac{1}{2}(c_x - b_x)^2 + \frac{1}{2}(c_y - b_y)^2. \tag{28}$$

In addition to the thickness constraint Eq. (22), we also need to preserve the force balance we reached in Step 1. This requires the weight of the hollowed object to be kept unchanged, which equals a vanishing Eulerian semiderivative of $I_{(\rho_s;\Omega)}$

$$\mathrm{d}I_{(\rho_s;\Omega)} = \rho_s \int_\Gamma v_n(\boldsymbol{x})\mathrm{d}\Gamma = 0. \tag{29}$$

Although Eq. (29) is treated as a hard constraint at each iteration, the gravity of the hollowed model may still change a bit because of the numerical errors introduced in remeshing during surface advancing. We solve this problem by explicitly compensating such errors (similar to post-stabilization [41]). Let G_0 be the gravity of the hollowed object at the beginning of an iteration, we require the change of gravity equals the difference between B and G_0

$$\mathrm{d}I_{(\rho_s;\Omega)} g\Delta t = B - G_0, \tag{30}$$

which finally turns to

$$\mathrm{d}I_{(\rho_s;\Omega)} = \rho_s \int_\Gamma v_n(\boldsymbol{x})\mathrm{d}\Gamma = \frac{B - G_0}{g\Delta t}. \tag{31}$$

An example of the result after the second step is shown in Fig. 4(d).

Step 3 Aligning the center of mass with the center of buoyancy is not sufficient to ensure the stability of floating object by itself. The concept relating to the stability of a floating object is so-called metacenter, which is a point directly above the center of buoyancy.[1)] A floating object is stable if its center of mass is below the metacentre (but the center of mass can be above the center of buoyancy). Note that only for very simple geometries, an analytic representation about the metacenter exists. Therefore, as the third step, we choose to lower the center of mass as much as possible. It largely increases the chance the object floats stably in the water.

The objective function of Step 3 is the z component of the center of mass

$$J_{stable} = c_z. \tag{32}$$

Similar to Step 2, we need to preserve the results of previous steps. In addition to the thickness constraint Eq. (22) and the gravity constraint Eq. (31), we add two more constraints.

$$\mathrm{d}c_x = \frac{b_x - c_x^0}{\Delta t} \quad \text{and} \quad \mathrm{d}c_y = \frac{b_y - c_y^0}{\Delta t}. \tag{33}$$

Again we compensate the numerical error here, and c_x^0 and c_y^0 are the x and y components of the center of mass at the beginning of the iteration. An example of the final result after Step 3 is shown in Fig. 4(e).

4.3 Design of spinnable objects

The computational design of spinnable objects has been investigated recently [2]. Their method optimizes the interior shape to adjust an object's rotational dynamics property using an adaptive voxel grid. Our algorithm can also be used to design spinnable objects such as spinning tops, while resulting in a smooth interior shape.

[1)] The exact position of metacentre is determined by the outer shape and immersion height of the object, and changes with the angle of heel of the object from the original orientation. In an equilibrium situation where the center of mass is aligned vertically with the center of buoyancy, the metacentre is directly above the center of buoyancy by a distance $d = \frac{I}{V}$ where I is the moment of inertia of the plane of floatation with respect to a horizontal axis and V is the immersion volume. Only for very simple geometries, d can be expressed as an analytic function of the angle of heel. For more details, please refer to Mégel and Kliava [42]

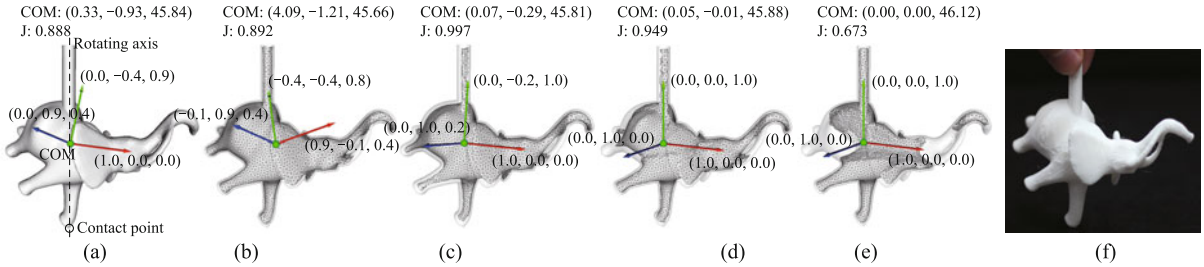


Fig. 5 The workflow of spinning application (The blue point is the ground contact point and is set as the origin of the world coordinate system. J is the spinnability energy computed by using Eq. (38)). (a) Input; (b) initial interior shape; (c) after aligning COM; (d) after aligning principal axes; (e) after minmizing spinnablity; (f) print

The input is a triangle mesh representing the outer surface of an object with its lowest point as the ground contact point, as shown in Fig. 5(a). The origin of the world coordinate system is located at the ground contact point, and the z-axis is expected to be the rotating axis. The rotational dynamics property of a rigid object is described by its inertia tensor

$$\boldsymbol{I} = \begin{pmatrix} s_{y^2} + s_{z^2} - \dfrac{{s_z}^2}{s_1} & -s_{xy} & -s_{xz} \\ -s_{xy} & s_{x^2} + s_{z^2} - \dfrac{{s_z}^2}{s_1} & -s_{yz} \\ -s_{xz} & -s_{yz} & s_{x^2} + s_{y^2} \end{pmatrix}. \tag{34}$$

Similar to the floating application, we divide the optimization into three steps. In each step, we formulate the same objective function as proposed by Bächer et al. [2], but use a surface discretization instead.

Step 1 To make a model spinnable, we first align the center of mass with the rotating axis, which means vanishing the x and y components of the center of mass. Thus the objective function of the first step is

$$J_{com} = \frac{1}{2}(c_x)^2 + \frac{1}{2}(c_y)^2. \tag{35}$$

Only the thickness constraint Eq. (22) needs to be satisfied here. A result of this step can be seen in Fig. 5(c).

Step 2 We then align the maximal principal axis with the rotating axis

$$J_{axis} = \frac{1}{2}\left(\frac{s_{xz}}{s_{x^2} + s_{y^2}}\right)^2 + \frac{1}{2}\left(\frac{s_{yz}}{s_{x^2} + s_{y^2}}\right)^2. \tag{36}$$

Meanwhile we use constraints to preserve the aligned center of mass

$$\mathrm{d}c_x = -\frac{c_x^0}{\Delta t} \quad \text{and} \quad \mathrm{d}c_y = -\frac{c_y^0}{\Delta t}. \tag{37}$$

As mentioned above, we use c_x^0 and c_y^0 instead of 0 on the right hand side to compensate the numerical errors. Figure 5(d) shows a result after this step.

Step 3 After both the center of mass and the maximal principal axis are aligned, we minimize the spinnability energy of the object

$$J_{spin} = \frac{1}{2}\left(\frac{I_a}{I_c}\right)^2 + \frac{1}{2}\left(\frac{I_b}{I_c}\right)^2, \tag{38}$$

where I_a, I_b and I_c are the three principal moments of inertia. As the maximal principal axis has already been aligned with the rotating axis, $I_c = s_{x^2} + s_{y^2}$. I_a and I_b are the two eigenvalues of the top-left 2×2 sub-matrix of $\boldsymbol{I}$ and can be computed analytically as

$$I_a, I_b = \frac{T}{2} \pm \left(\frac{T^2}{4} - D\right)^{1/2}, \tag{39}$$

where

$$\begin{aligned} T &= s_{xx} + s_{yy} + 2s_{zz} - 2\frac{{s_z}^2}{s_1}, \\ D &= \left(s_{xx} + s_{zz} - \frac{{s_z}^2}{s_1}\right)\left(s_{yy} + s_{zz} - \frac{{s_z}^2}{s_1}\right) - {s_{xy}}^2. \end{aligned} \tag{40}$$

The Eulerian semiderivative of Eq. (38) is derived in Appendix B. In addition to the thickness constraint Eq. (22) and the center of mass preserving constraint Eq. (37), we add two more constraints to preserve the aligned maximal principle axis

$$\mathrm{d}\left(\frac{s_{xz}}{s_{x^2} + s_{y^2}}\right) = -\frac{s_{xz}^0}{s_{x^2}^0 + s_{y^2}^0}/\Delta t, \tag{41}$$

$$\mathrm{d}\left(\frac{s_{yz}}{s_{x^2} + s_{y^2}}\right) = -\frac{s_{yz}^0}{s_{x^2}^0 + s_{y^2}^0}/\Delta t, \tag{42}$$

where s_α^0 is the value of s_α at the beginning of the iteration. A result after all three steps is shown in Fig. 5(e).

4.4 Design of statically balanced objects

We also apply our algorithm to the static object balancing problem. This application is previously addressed by Prévost et al. [1]. Their method adopts the voxelization representation and adjusts the center of mass in a heuristic way.

As shown in Fig. 6(a), we generate a convex hull from the ground contact points of the input object and set the center

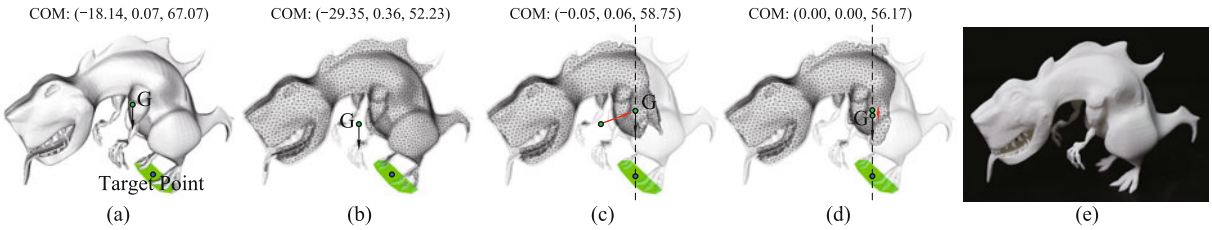


Fig. 6 The workflow of static balancing application (The green zone is the convex hull generated from the ground contact points of the object. The blue point is the center of the convex hull, and is set as the target point and the origin of the world coordinate system). (a) Input; (b) initial interior shape; (c) after aligning COM; (d) after lowering down COM; (e) print

of the convex hull as the target point $\boldsymbol{p}$. The optimization is divided into two steps.

Step 1 To enable an object to stand on the ground plane, we first move the projection of the center of mass on the ground toward the target point $\boldsymbol{p}$

$$J_{align} = \frac{1}{2}(c_x - p_x)^2 + \frac{1}{2}(c_y - p_y)^2. \tag{43}$$

Only the thickness constraint Eq. (22) needs to be satisfied here. A result of this step can be seen in Fig. 6(c).

Step 2 To make the object stand as stably as possible, we lower down the center of mass

$$J_{stable} = c_z, \tag{44}$$

while preserving the x and y components of the center of mass as

$$\mathrm{d}c_x = \frac{p_x - c_x^0}{\Delta t} \quad \text{and} \quad \mathrm{d}c_y = \frac{p_y - c_y^0}{\Delta t}. \tag{45}$$

An example of the result can be seen in Fig. 6(d).

Prévost et al. [1] generously made their 3D models publicly available, allowing us to repeat their examples using our method. Thanks to the continuous representation of the carved shape and its deformation, we find that our carving method is more effective. For the example shown in Fig. 5, as reported in their paper, their discretized carving method cannot completely balance the object. In contrast, our method successfully finds interior carving to achieve static balance (see Fig. 6(e) and the video).

5 Results

We implemented our algorithm on a desktop PC equipping an Intel i7-4770 3.4 Ghz CPU and 16GB RAM. All computations are executed in a single thread. All our models have size 10–20 cm. We set the density of the printing material as 1 200 kg/m^3 and the density of the fluid as 1 000 kg/m^3 (water). For "fish", "elephant" and "ellipsoid", we use 1.0 mm as the minimal printing thickness; for others we use 1.5 mm. The termination threshold ϵ in Algorithm 1 is set as 1e–5.

All the experimental results are printed by a low-cost FDM 3D printer using PLA materials. To facilitate the removing of the supporting structures inside the models, we manually cut the mesh into individual pieces and glue the printed pieces together. For the floatation application, we smear Vaseline over the cracks between the pieces to make the model watertight. Compared with the self-weight of the models, the mass influence of the glue and Vaseline is negligible.

Floating objects design Figures 1(c) and 4(f) show a result of the floatation application, a bear on an iceberg. Our method hollows the interior of the model to decrease its volume from 313 853.3 mm^3 to 82 163.5 mm^3, while the target volume is 82 161.9 mm^3 which is determined by the buoyancy force and printing material density. The center of mass is adjusted from (8.071, –1.868, 25.766) to (0.002, 0.001, 20.546) (in unit of millimeter, mm, and the origin of the world coordinate system is put at the center of buoyancy). From these numbers, we can see that the constraints of the volume and the center of mass are both accurately satisfied. As shown in the figures, the immersion height and the pose of the printed model in the water closely match the design.

Figures 7 and 8 show another two examples of the floatation application. Both models have large parts above the

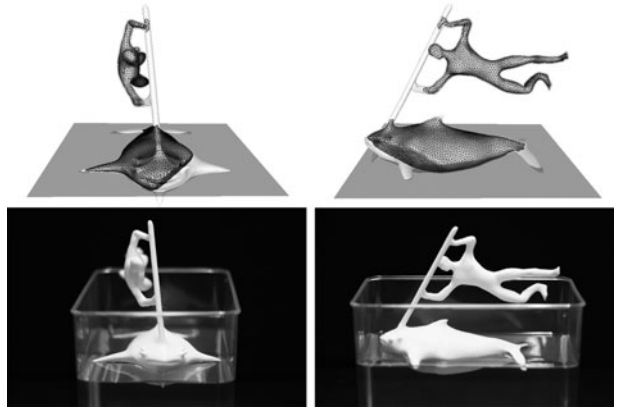


Fig. 7 Floating object design: fish

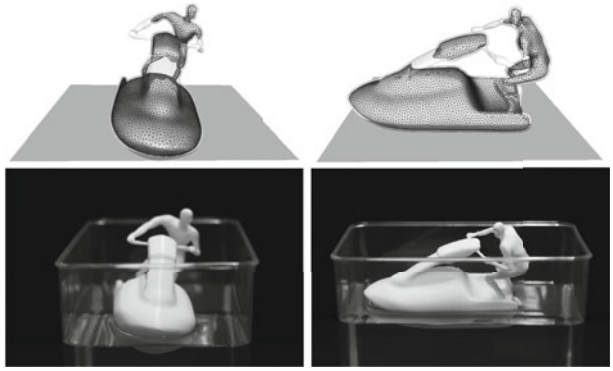


Fig. 8 Floating object design: motorboat

water-level and cannot float with the specified immersion height and pose before optimization. Our method accurately adjusts their volumes and center of mass such that the optimized results can float stably in water with the designed poses.

For all the three models, our optimizer converges in 1–2 minutes. The detailed performances are shown in Table 1. The three size columns are respectively the sizes of the input triangle mesh, the initial interior boundary mesh and the interior boundary mesh after optimization. In the performance columns, the time in the column Init includes computing the buoyancy force and building the Kd-tree for thickness constraints. The following four columns are the performances of the four steps in the optimization: shrinking the outer surface to obtain initial interior shape, adjusting volume to balancing gravity and buoyancy force, aligning the center of mass and lowering down the center of mass respectively. The last column is the percentage of time costed in surface tracking. The convergence curve of the bear model is shown in Fig. 9. From the table and the curve, we can see that the last optimization step needs more iterations than the previous two steps and thus costs most of the computing time. The reason is that the last step has more constraints than previous two steps, and the interior boundary advances more elaborately.

Spinnable objects design Figure 5(f) shows an elephant-shaped spinning top. Before optimization, the center of mass of the input model is not directly above the contact point and the maximal principal axis is misaligned with the rotating axis (z axis). Our algorithm generates a multi-void interior shape to align the center of mass and the maximal principal axis exactly, and maximizes its spinnability. Twisted by fingers, the printed model keeps spinning for a while, as recorded in the accompanied video. The whole computation is completed in less than 1 minute. Table 2 shows the detailed performance. The time in the column Init includes building the Kd-tree for thickness constraints. The following four columns are the performances of the four steps in the optimization: shrinking outer surface to obtain the initial interior shape, aligning the center of mass, aligning the principal axes and minimizing the spin energy respectively. The convergence curve of the optimization is shown in Fig. 10.

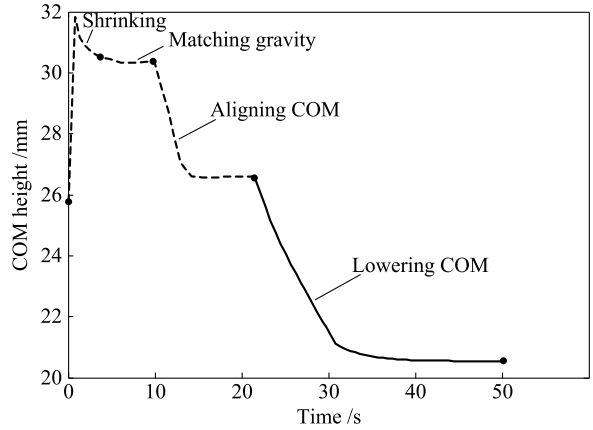


Fig. 9 The convergence curve of the bear model (The abscissa is the computing time. The ordinate is the height of the center of mass above the center of the buoyancy (Eq. (32)). The end of each step is marked as a red point on the curve. The first three steps (dotted) shrink the outer surface to obtain the initial interior boundary and advance it to satisfy the constraints, and the last step (solid) minimizes the height of the center of mass.)

Figure 11 shows the controllability we have on the smoothness of the interior shape's boundary. The result in Fig. 11(a) is computed without the smoothness term. The interior boundary has sharp edges but the smallest spin energy. In Fig. 11(b) we add the smoothness term Eq. (24) to the objective function Eq. (38) with a small weight, 1e–7. The sharp edges of the interior boundary are smoothed while the overall shape is preserved, and the spin energy is almost unaffected. In Fig. 11(c) we use a weight 5 times larger. The shape of the interior boundary becomes even smoother while the spin energy slightly increases. Adjusting the weight of the smoothness energy term provides us with a way to balance the smoothness

Table 1 Performance of the floatation application

Model	Size (# vertex / # triangles)			Performance (# iteration / time (seconds))						
	Input mesh	Initial surf	Result surf	Init	Shrink	Gravity	Align COM	Lower COM	Total	Surf/%
Bear	3 993/7 990	5 217/10 438	4 076/8 156	5.2	6/3.3	8/6.3	16/11.4	50/29.5	55.8	60.6
Fish	7 500/15 000	7 710/15 420	5 851/11 690	7.2	4/4.0	6/10.4	20/27.7	29/35.1	84.3	46.0
Motorboat	7 496/15 000	8 141/16 290	6 673/13 324	4.4	6/6.2	7/12.9	6/11.7	44/62.8	97.0	43.3

Table 2 Performance of the spinning application

Model	Size (# vertex/# triangles)			Performance (# iteration/time (seconds))						
	Input mesh	Begin surf	End surf	Init	Shrink	Align COM	Align axis	Spinability	Total	Surf/%
Elephant	4 416/8 840	5 517/11 042	3 126/6 240	2.2	4/2.7	8/8.0	7/6.7	42/30.2	49.9	46.9
Teapot	734/1 468	3 900/7 800	2 879/5 750	2.5	6/2.4	5/2.6	3/1.5	83/29.3	38.4	65.8
Ellipsoid	2 526/5 048	2 623/5 242	2 402/4 800	1.2	4/1.0	5/2.6	6/3.0	85/37.7	45.4	46.2

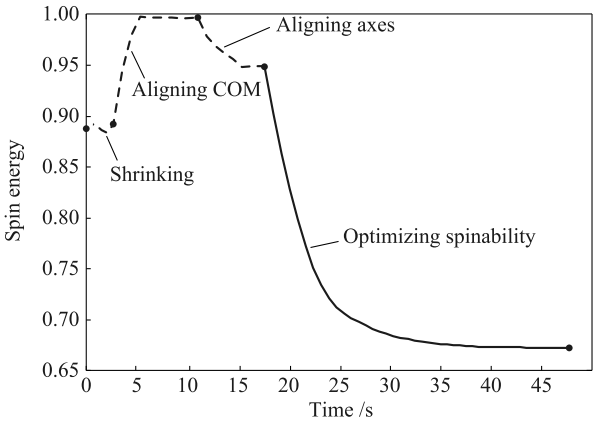


Fig. 10 Convergence curve of the elephant model (The abscissa is the computing time. The ordinate is the spin energy (Eq. (38)). The end of each step is marked as a red point on the curve. The first three steps (dotted) shrink the outer surface to obtain the initial interior boundary and advance it to satisfy the constraints, and the last step (solid) minimizes the spin energy.)

of the interior shape against the spinnability of the result. This comparison demonstrates the controllability of our method on the surface properties of the interior boundary, which is absent in voxelization-based methods [1,2].

Statically balanced objects design Figure 6(e) and Fig. 12 show two results of the static balancing application. Both models fall on the ground before optimization and stand stably after the interior shapes computed by our method are hollowed. For the dinosaur model, the center of mass is first aligned vertically to the target point with an error smaller than 0.001 mm and then moved as low as possible. For the spheres model, the center of mass cannot be adjusted to a position directly above the target point because of the minimal wall thickness requirement. Our algorithm thus moves it as close as possible to the target point horizontally. The second optimization step stops after only six iterations as there is no more deformation room for the interior shape to lower down the center of mass. The performance of both results is reported in Table 3. Compared with the results of previous applications, the numbers of iterations to align the center of mass are much larger. In these two models, the initial centers of mass are much farther from the target points, and hence the interior shapes need to change more to adjust the center of mass. Please see the video for a complete optimization process.

Comparison with spin-it We compare our method with the hollowing part of the spin-it method [2] which adopts voxelization as the discretization method and is regarded as the state-of-the-art. The nonlinear optimization problem of spin-it method is solved with Knitro [43] library by using the SLQP algorithm. All calculations are executed in single thread. We run spin-it method many times to experiment different maximal refinement levels, starting from level 7 as the initial refinement level. The iterating is terminated when there are no voxels can be further split or the relative change of the energy is less than the same convergence threshold (ϵ =1e–5) used in our method.

Figure 13 compares the results of spin-it and our method on the Ellipsoid model of the spinning application. From the figure we can see that the interior voids of the two results have similar overall shapes. But the smooth surface of our result closely matches the outer surface and generates walls as thin as the minimal wall thickness, while the result of spin-it can only approximate the outer surface with small voxels. Table 4 compares the performances of two methods. The max level column is the maximal boundary/interior refinement level.

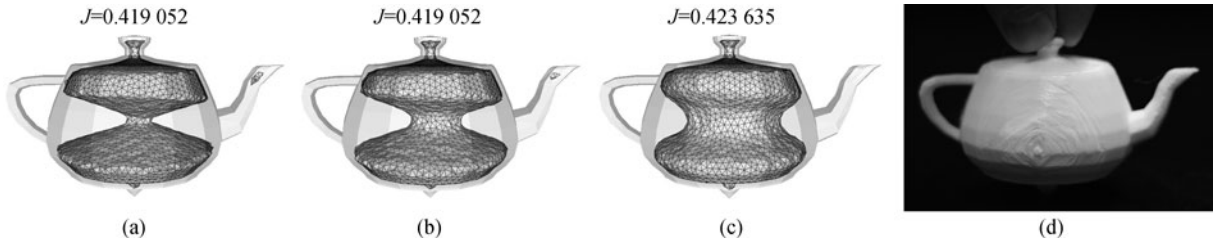


Fig. 11 Comparison of different smoothness energy term weights (w is the weight of the smoothness energy term. J is the spinnability energy in Eq. (38).). (a) w = 0.0; (b) w =le–7; (c) w =5e–7; (d) print

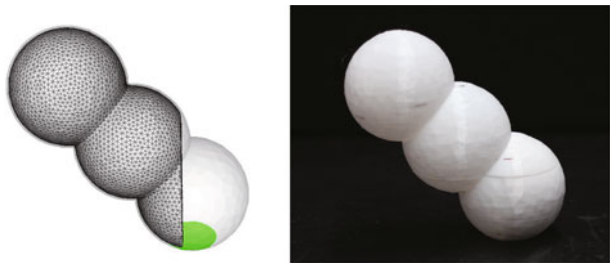


Fig. 12 Statically balanced object design: spheres

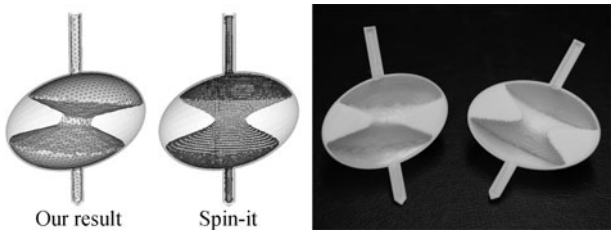


Fig. 13 Comparison of result meshes with spin-it [2] (The left two pictures are the optimized interior shapes of our method and spin-it. The right picture is the photo of the real objects printed with the two results.)

The #Voxel is the number of voxels in the octree. The #DOF column is the number of DOF before/after optimization. The spin energy is computed as Eq. (38). From the table we can see that although spin-it can obtain results faster using coarse refinements, to obtain a result comparable to ours it needs an impractical high refinement level. As in the third row, to obtain the spinning energy 0.517, we must set its maximal refinement level to 11/10, but the optimizer fails to return a solution in a few minutes during the third iteration because 339 686 DOFs have been generated. And the time cost of tree-building plus the first two iterations have already exceeded 90 seconds. In contrast, our method gets a spinning energy 0.514 using only 2 402 DOFs in 46 seconds. High refinement level also leads to voxels of prohibitive small size, e.g., 0.230 mm for a 10-level voxel and 0.115 mm for a 11-level voxel, which could not be fabricated properly by low-cost FDM printers.

We also compare spin-it method with our method on the bear model for the floatation application and on the dinosaur model for the statically balancing application, as summarized in Tables 5 and 6. Again, from these results we can see that to obtain a result with comparable accuracy, spin-it method must use a very high refinement level which leads to long computing time and excessively small voxels. For the bear model, spin-it method even fails to generate a result satisfying all constraints when a relatively coarse refinement level is used (first two rows). For the dinosaur model, with a coarse refinement level (the first three rows) spin-it method stops the optimization after the first iteration as all voxels are either fully filled or hollowed and thus no new voxels are split. And even refining the octree to level 12 which leads to 9.5 million voxels and 0.088 mm voxel size, spin-it method still cannot adjust the center of mass directly above the target point. Our method not only adjusts the center of mass directly above the target point but also lowers it down to make the model stand more stably, using much less DOFs.

Discussion and limitations The result of our method is affected by the initial shape of the interior voids. In our experiments we find that shrinking the outer surface is a good choice to obtain the initial interior void, especially for models in the floatation application where most interior volumes of the objects need to be hollowed.

As shown in the performance tables, the computation time spent on each iteration is related to the mesh resolution of the interior boundary surface (note that half the time is spent on surface tracking). The number of iterations needed for convergence varies from model to model, but in general we find that it depends on to what extent the interior void needs to change its shape. The larger and more complex the change is, the more iterations are needed.

Due to the printing accuracy, the density of the printed ob-

Table 3 Performance of the static balancing application

Model	Size (# vertex / # triangle)			Performance (# iteration / time (seconds))					
	Input mesh	Initial surf	Result surf	Init	Shrink	Align COM	Lower down COM	Total	Surf/%
Dinosaur	5 047/10 090	6 629/13 254	3 618/7 228	2.7	6/4.8	64/54.0	110/73.7	135.2	47.9
Spheres	1 441/2 878	6 242/12 480	4 822/9 640	1.7	6/3.9	136/93.0	6/4.7	103.3	70.5

Table 4 Comparison with spin-it [2] on the Ellipsoid model for the spinning application

	Max level	# Voxel	# DOF	Spin energy	Build tree time/s	Optimization time/s
Spin-it	9/8	180 720	15 141/22 895	0.549	4.4	4.4
	10/9	695 080	15 830/77 204	0.527	18.1	12.4
	11/10	2 749 489	16 147/>339 686	0.517	72.6	>21.6
Our	-	-	2 526/2 402	0.514	-	45.4

Note: In the third row of spin-it method, ">" means "larger than" as too many DOFs are generated as the result of node splitting thus we have to stop the optimization after 2 iterations

Table 5 Comparison with spin-it [2] on the bear model for the flotation application

	Max level	# Voxel	# DOF	COM height/mm	Build tree time/s	Optimization time/s
Spin-it	9/8	232 478	13 354/-	n/a	6.1s	n/a
	10/9	915 279	14 163/-	n/a	25.8s	n/a
	11/10	3 651 754	14 602/63 033	23.037	100.7s	48.4s
	12/11	14 609 204	14 826/29 062	21.156	408.8s	24.0s
Our	-	-	3 993/4 076	20.546	-	55.8s

Note: In the 1st two rows, the non-linear solver of spin-it method failed to find a feasible solution satisfying the COM constraints and gravity constraints simultaneously

Table 6 Comparison with spin-it [2] on the dinosaur model for the statically balancing application

	Max level	# Voxel	# DOF	COM distance/mm	COM height/mm	Build tree time/s	Optimization time/s
Spin-it	9/8	148 317	6 198/6 198	3.210	60.738	4.0	0.4
	10/9	591 998	6 750/6 750	1.613	59.638	17.5	1.2
	11/10	2 379 245	7 112/7 112	0.656	59.402	72.1	2.5
	12/11	9 582 504	7 282/>75 645	0.133	59.077	286.0	>699.9
Our	-	-	5 047/3 618	0.0	56.170	-	135.2

Note: In the 4th row of spin-it method, ">" means "larger than" as spin-it method does not converge after more than 10 minutes so we stop the optimization

ject varies a bit from the density of the printing material. In our floatation application, such density difference could lead to different immersion height, buoyancy force and center of buoyancy position, which finally makes the printed object float in a pose slightly different from the design.

For the floatation application, although our optimization largely increases the floatation possibilities and works well in practice for all our models, it cannot fully guarantee the stability of the optimized result. This is due to a fact that the exact position of the metacentre changes with the orientation of the model in the fluid and is hard to calculate. Incorporating a good approximation for metacenter computation could be an interesting direction to explore.

6 Conclusion

In this paper, we propose a general optimization algorithm for interior carving in 3D fabrication. Unlike previous methods, we use an explicit triangular mesh to represent the carving boundary of the interior voids. We use Eulerian semiderivative to relate the time derivative of the objective function to a virtual velocity field. In each iteration of the optimization, we solve a linear programming problem to find the velocity field that decreases the objective function fastest, and then advance the boundary surface in this velocity field using surface tracking. We apply the general optimization algorithm to a novel application for floating object design and two other perviously investigated applications. To verify the effectiveness of our algorithm, we 3D print the optimized objects and compare them with the digital designs.

Acknowledgements We would like to thank the reviewers for their constructive comments. Xiang Chen is partially supported by NSFC (Grant No. 61303136) and the Fundamental Research Funds for the Central Universities. Kun Zhou is partially supported by NSFC (Grant No. 61272305) and National Program for Special Support of Eminent Professionals of China.

Appendixes

Appendix A Gradient descent direction of $\boldsymbol{v}(\boldsymbol{x})$

Here we consider a normalized velocity field $v_n(\boldsymbol{x})$ such that

$$\int_{\Gamma_0} v_n(\boldsymbol{x})^2 \mathrm{d}\Gamma = 1. \tag{46}$$

We seek a velocity $v_n(\boldsymbol{x}), \forall \boldsymbol{x} \in \Gamma_0$ under this constraint to minimize the Eulerian derivative value Eq. (12) of the objective function. Particularly, we solve the constrained optimization problem,

$$\min_{v_n} \int_{\Gamma_0} P(\boldsymbol{x}) v_n(\boldsymbol{x}) \mathrm{d}\Gamma, \text{ s.t. the unit constraint Eq. (46).}$$

In this case, the optimal velocity $v_n(\boldsymbol{x})$ can be solved explicitly using the method of Lagrange multipliers and the Euler-Lagrange equation. The resulting optimal $v_n(\boldsymbol{x})$ is indeed proportional to $-P(\boldsymbol{x})$,

$$v_n(\boldsymbol{x}) = -\frac{1}{(\int_{\Gamma_0} P(\boldsymbol{x})^2 \mathrm{d}\Gamma)^{\frac{1}{2}}} P(\boldsymbol{x}). \tag{47}$$

Appendix B Eulerian semiderivatives

Common terms One term we frequently use in all applications is

$$s_f = I_{(\rho_s f;V)} - I_{(\rho_s f;\Omega)}, \tag{48}$$

whose Eulerian semiderivative is

$$\mathrm{d}s_f = -\mathrm{d}I_{(\rho_s f;\Omega)} = -\int_{\Gamma} f(\boldsymbol{x}) v_n(\boldsymbol{x}) \mathrm{d}\Gamma. \tag{49}$$

The component of the center of mass are also frequently used

$$c_\alpha = \frac{s_\alpha}{s_1}, \quad \text{where } \alpha = x, y, z, \tag{50}$$

whose Eulerian semiderivatives can be computed using the chain rule

$$\mathrm{d}c_\alpha = \frac{\mathrm{d}s_\alpha s_1 - \mathrm{d}s_1 s_\alpha}{{s_1}^2}. \tag{51}$$

With these basic terms, the Eulerian semiderivatives of most objective functions and constraints formulated in Section 4 can be derived using the chain rule in a straightforward manner. The only nontrivial one is Eq. (38), whose Eulerian derivative is derived below.

Eulerian semiderivative of Eq. (38) Applying the chain rule to Eq. (38), we obtain

$$\mathrm{d}J_{spin} = \frac{I_a}{I_c^2}\mathrm{d}I_a + \frac{I_b}{I_c^2}\mathrm{d}I_b - \frac{I_a^2 + I_b^2}{I_c^3}\mathrm{d}I_c. \tag{52}$$

Then, by applying the chain rule to Eq. (39) and Eq. (40), the $\mathrm{d}I_a$ and $\mathrm{d}I_b$ can be computed as

$$\mathrm{d}I_a, \mathrm{d}I_b = (\frac{1}{2} \pm \frac{T}{4}(\frac{T^2}{4} - D)^{-\frac{1}{2}})\mathrm{d}T \mp \frac{1}{2}(\frac{T^2}{4} - D)^{-\frac{1}{2}}\mathrm{d}D, \tag{53}$$

where

$$\mathrm{d}T = \mathrm{d}s_{xx} + \mathrm{d}s_{yy} + 2\mathrm{d}s_{zz} - \frac{4s_z\mathrm{d}s_z s_1 - 2s_z^2\mathrm{d}s_1}{s_1^2}, \tag{54}$$

and

$$\begin{aligned}\mathrm{d}D = &(\mathrm{d}s_{xx} + \mathrm{d}s_{zz} - \frac{2s_z\mathrm{d}s_z s_v - s_z^2\mathrm{d}s_v}{s_v^2})(s_{yy} + s_{zz} - \frac{s_z^2}{s_v}) \\ &+(\mathrm{d}s_{yy} + \mathrm{d}s_{zz} - \frac{2s_z\mathrm{d}s_z s_v - s_z^2\mathrm{d}s_v}{s_v^2})(s_{xx} + s_{zz} - \frac{s_z^2}{s_v}) \\ &-2s_{xy}\mathrm{d}s_{xy}.\end{aligned} \tag{55}$$

Finally the $\mathrm{d}I_c$ can be derived by applying the chain rule again

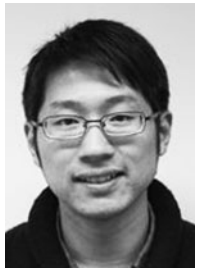
$$\mathrm{d}I_c = \mathrm{d}s_{x^2} + \mathrm{d}s_{y^2}. \tag{56}$$

References

1. Prévost R, Whiting E, Lefebvre S, Sorkine-Hornung O. Make it stand: balancing shapes for 3D fabrication. ACM Transactions on Graphics, 2013, 32(4): 81
2. Bächer M, Whiting E, Bickel B, Sorkine-Hornung O. Spin-it: optimizing moment of inertia for spinnable objects. ACM Transactions on Graphics, 2014, 33(4): 96
3. Christiansen A N, Schmidt R, Bærentzen J A. Automatic balancing of 3D models. Computer-Aided Design, 2015, 58: 236–241
4. Chen S, Torterelli D. Three-dimensional shape optimization with variational geometry. Structural Optimization, 1997, 13(2): 81–94
5. Braibant V, Fleury C. Shape optimal design using B-splines. Computer Methods in Applied Mechanics and Engineering, 1984, 44(3): 247–267
6. Xu D, Ananthasuresh G K. Freeform skeletal shape optimization of compliant mechanisms. Journal of Mechanical Design, 2003, 125(2): 253–261
7. Bendsoe M P. Optimal shape design as a material distribution problem. Structural Optimization, 1989, 1(4): 193–202
8. Wang M Y, Wang X M, Guo D M. A level set method for structural topology optimization. Computer Methods in Applied Mechanics and Engineering, 2003, 192(1): 227–246
9. Zhou M, Pagaldipti N, Thomas H, Shyy Y. An integrated approach to topology, sizing, and shape optimization. Structural and Multidisciplinary Optimization, 2004, 26(5): 308–317
10. Haftka R T, Grandhi R V. Structural shape optimizationa——a survey. Computer Methods in Applied Mechanics and Engineering, 1986, 57(1): 91–106
11. Saitou K, Izui K, Nishiwaki S, Papalambros P. A survey of structural optimization in mechanical product development. Journal of Computing and Information Science in Engineering, 2005, 5(3): 214–226
12. Luo L, Baran I, Rusinkiewicz S, Matusik W. Chopper: partitioning models into 3D-printable parts. ACM Transactions on Graphics, 2012, 31(6)
13. Attene M. Shapes in a box: disassembling 3D objects for efficient packing and fabrication. Computer Graphics Forum, 2015
14. Zhou Y B, Sueda S, Matusik W, Shamir A. Boxelization: folding 3D objects into boxes. ACM Transactions on Graphics, 2014, 33(4): 71
15. Bickel B, Kaufmann P, Skouras M, Thomaszewski B, Bradley D, Beeler T, Jackson P, Marschner S, Matusik W, Gross M. Physical face cloning. ACM Transactions on Graphics, 2012, 31(4): 118
16. Skouras M, Thomaszewski B, Coros S, Bickel B, Gross M. Computational design of actuated deformable characters. ACM Transactions on Graphics, 2013, 32(4): 82
17. Chen X, Zheng C, Xu W, Zhou K. An asymptotic numerical method for inverse elastic shape design. ACM Transactions on Graphics, 2014, 33(4): 95
18. Bächer M, Bickel B, James D L, Pfister H. Fabricating articulated characters from skinned meshes. ACM Transactions on Graphics, 2012, 31(4): 47
19. Calì J, Calian D A, Amati C, Kleinberger R, Steed A, Kautz J,Weyrich T. 3D-printing of non-assembly, articulated models. ACM Transactions on Graphics, 2012, 31(6): 130
20. Zhu L, Xu W, Snyder J, Liu Y, Wang G, Guo B. Motion-guided mechanical toy modeling. ACM Transactions on Graphics, 2012, 31(6): 127
21. Coros S, Thomaszewski B, Noris G, Sueda S, Forberg M, Sumner R W, Matusik W, Bickel B. Computational design of mechanical characters. ACM Transactions on Graphics, 2013, 32(4): 83
22. Umetani N, Igarashi T, Mitra N J. Guided exploration of physically valid shapes for furniture design. ACM Transactions on Graphics, 2012, 31(4): 86
23. Vouga E, Höbinger M, Wallner J, Pottmann H. Design of self-supporting surfaces. ACM Transactions on Graphics, 2012, 31(4): 87
24. Panozzo D, Block P, Sorkine-Hornung O. Designing unreinforced masonry models. ACM Transactions on Graphics, 2013, 32(4): 91
25. De Goes F, Alliez P, Owhadi H, Desbrun M. On the equilibrium of simplicial masonry structures. ACM Transactions on Graphics, 2013, 32(4): 93
26. Wang W, Wang T Y, Yang Z, Liu L, Tong X, Tong W, Deng J, Chen F, Liu X. Cost-effective printing of 3D objects with skin-frame structures. ACM Transactions on Graphics, 2013, 32(6): 177
27. Lu L, Sharf A, Zhao H,Wei Y, Fan Q, Chen X, Savoye Y, Tu C, Cohen-

Or D, Chen B. Build-to-last: strength to weight 3D printed objects. ACM Transactions on Graphics, 2014, 33(4): 97

28. Stava O, Vanek J, Benes B, Carr N, Měch R. Stress relief: improving structural strength of 3D printable objects. ACM Transactions on Graphics, 2012, 31(4): 48
29. Zhou Q, Panetta J, Zorin D. Worst-case structural analysis. ACM Transactions on Graphics, 2013, 32(4): 137
30. Xie Y, Xu W, Yang Y, Guo X, Zhou K. Agile structural analysis for fabrication-aware shape editing. Computer Aided Geometric Design, 2015, 35: 163–179
31. Musialski P, Auzinger T, Birsak M, Wimmer M, Kobbelt L. Reduced-order shape optimization using offset surfaces. ACM Transactions on Graphics, 2015, 34(4): 102
32. Delfour M C, Zolésio J P. Shapes and Geometries: Metrics, Analysis, Differential Calculus, and Optimization. Philadelphia: Siam, 2011
33. Brakke K A. The surface evolver. Experimental Mathematics, 1992, 1(2): 141–165
34. Sethian J A. Level set methods and fast marching methods. Journal of Computing and Information Technology, 2003, 11(1): 1–2
35. Nesterov Y, Nemirovskii A. Interior-Point Polynomial Algorithms in Convex Programming. Philadelphia: Siam, 1994
36. Makhorin A, Andrew O. GLPK (GNU linear programming kit). 2008
37. Osher S, Fedkiw R. Level set methods and dynamic implicit surfaces. Surfaces, 2002, 44
38. Brochu T, Bridson R. Robust topological operations for dynamic explicit surfaces. SIAM Journal on Scientific Computing, 2009, 31(4): 2472–2493
39. Sorkine O, Cohen-Or D. Least-squares meshes. In: Proceedings of Shape Modeling Applications. 2004
40. McGrail S. Boats of the World: from the Stone Age to Medieval Times. New York: Oxford University Press, 2004
41. Ascher U M, Chin H, Reich S. Stabilization of DAEs and invariant manifolds. Numerische Mathematik, 1994, 67(2): 131–149
42. Mégel J, Kliava J. Metacenter and ship stability. American Journal of Physics, 2010, 78(7): 738–747
43. Byrd R H, Nocedal J, Waltz R A. Knitro: an integrated package for nonlinear optimization. In: Di Pillo G, Roma M, eds. Large-Scale Nonlinear Optimization, Vol 83. Springer, 2006, 35–59



Yue Xie received his BS in computer science from Jiangnan University, China in 2008. He is currently a PhD student in computer science at Zhejiang university, China. His research focuses on computer-aided design.

Ye Yuan received his BS in computer science from Zhejiang University, China in 2015. From September 2015, He became a master student in computer science at Carnegie Mellon University, USA. His research interests include physically based simulation, rendering and animation.

Xiang Chen received his PhD in computer science from Zhejiang University (ZJU), China in 2012. He is currently an assistant professor in the College of Computer Science and Technology, ZJU. His research interests include fabrication-aware design, image analysis/editing, shape modeling/retrieval and computer-aided design.

Changxi Zheng received his PhD in computer science from Cornell University, USA in 2012. He is currently an assistant professor in Computer Science Department in Columbia University, USA. His research interests include computer graphics, scientific computing and robotics.

Kun Zhou is a Cheung Kong Professor in the Computer Science Department of Zhejiang University (ZJU), China, and the director of the State Key Lab of CAD&CG, China. Prior to joining ZJU in 2008, he was a leader researcher of the Internet Graphics Group at Microsoft Research Asia, China. He received his BS degree and PhD degree in computer science from ZJU in 1997 and 2002, respectively. His research interests are in visual computing, parallel computing, human computer interaction, and virtual reality. He currently serves on the editorial/advisory boards of ACM Transactions on Graphics and IEEE Spectrum. He is a fellow of IEEE.