Fast Marching-Cubes-Style Volume Evaluation for Level Set Surfaces

Tetsuya Takahashi
Adobe

Christopher Batty
University of Waterloo

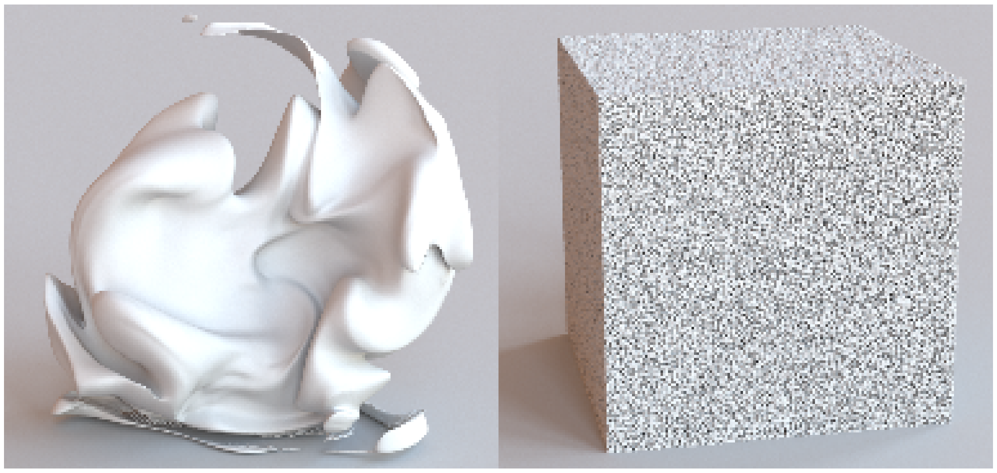

Figure 1. Left: A bunny shape represented by a grid-based signed distance field after being deformed by an incompressible curl-noise vector field. Right: The implied surface of a grid of randomly generated level set values in a cube domain. Our method efficiently evaluates the volume of these implicitly represented shapes *without building a mesh*, achieving faster performance over competing methods and generating volume results perfectly consistent with the explicitly reconstructed Marching Cubes meshes.

Abstract

We present an efficient and accurate volume evaluation method for grid-based level sets that computes the volume of the implicitly represented shape(s) in a manner consistent with Marching Cubes surface reconstruction. We utilize rotational symmetry to combine redundant Marching Cubes cases and avoid explicitly forming local triangulations using efficient volume computation formulae for pyramids and truncated prisms wherever possible, thereby achieving a fast and compact implementation. We demonstrate that our method is more efficient than previous approaches while generating results that converge with second-order accuracy and are perfectly consistent with volumes calculated directly from explicit Marching Cubes

meshes. We provide a full C++17 reference implementation to foster adoption of the proposed method (`https://github.com/tetsuya-takahashi/MC-style-vol-eval`).

1. Introduction

The level set method has been extensively used to perform numerical analysis on deforming, implicitly represented shapes (typically using a Cartesian grid) due to its various benefits, e.g., efficiency due to the regular grid structure and trivial handling of topological changes [Osher and Fedkiw 2004]. Another advantage is that marching-based surface reconstruction of level set data (e.g., using Marching Cubes (MC) [Lorensen and Cline 1987], Marching Tetrahedra (MT) [Doi and Koide 1991], or Marching Squares (MS) [Maple 2003]) is guaranteed to yield manifold, watertight meshes, except at the outer domain boundary. These features of the level set method have been critical in many applications, e.g., image processing, fluid simulation, and topology optimization.

In many such applications, the volume of a given shape is an important geometric quantity. However, the shape represented by the level set is modified at every step of the level set method, thus changing the volume. It is therefore essential to have a method that can accurately and efficiently evaluate level set volumes. An early approach [Kim et al. 2007; Bridson 2016] approximated the level set volume using 8-point quadrature on a Heaviside function applied to the level set (essentially supersampling with a one-level finer interpolated grid), but this approach tends to be much less accurate. Given the extensive use of MC to reconstruct surfaces, such as for rendering or other downstream processes, it is also natural to ask that the computed volumes be consistent with meshes reconstructed using standard MC. Although one could directly compute such consistent volumes *after* fully reconstructing the MC surface (e.g., using the usual triangle mesh volume calculation derived from the divergence theorem), this approach is typically rather slow; many of the steps required to build the triangle mesh are unnecessary if one desires only to compute volumes, as is the case in various applications.

A challenge in rapidly evaluating MC-consistent volumes without a full reconstruction is that, given each grid cell with eight level set corner values in 3D, we have 256 $(= 2^8)$ MC patterns, making it impractical to manually implement volume computations for each case. To avoid this issue, Min and Gibou [2007] and Bridson (within Batty's viscous fluid code [2013]) decomposed a cubic cell into multiple tetrahedra (each of which has only $16 = 2^4$ possible patterns) and individually evaluated their partial interior volumes with MT-style surfaces, summing up these volumes to get the total volume within the cubic cell. While this approach reduces implementation complexity, the decomposition into tetrahedra introduces directional bias and artificial bumps in the implied surface [Hansen and Johnson 2004], which can negatively affect the surface quality for applications, especially if voxel-wise volume plays a key role (e.g., [Bridson 2016]). In addition, the resulting volumes are inconsistent with meshes reconstructed based on MC. (For surface meshing, MC is often advantageous because it generates fewer triangles than MT, thus offering lower memory usage and greater rendering efficiency.)

Wang [2013] instead presented a method for computing consistent volumes based on MC without tetrahedral decomposition. This approach utilizes the rotational symmetry of each cubic grid cell to reduce the number of distinct MC cases and virtually reconstructs the closed triangle mesh for a given MC case before computing the corresponding volume using the divergence theorem. (By "virtually" we mean that, though no persistent mesh is built, sufficient vertex and triangle information is determined to enable the volume calculation.) Importantly, because volume calculation requires a closed volume, triangles must be formed not only for a cell's interior surface (as in MC), but also for any full or partial exterior faces of the cube deemed to be inside the material; thus, upward of a dozen triangles are often needed just for a single cell. We found that although this approach avoids fully reconstructing the surface, the virtual reconstruction of local MC triangulations is nevertheless costly. In addition, Wang [2013] neglected certain triangulation patterns that are needed to ensure proper consistency with MC tables (e.g., [Bourke 1994]).

In this paper, we present a more efficient, numerically verified, MC-style volume evaluation method. Our method utilizes rotational symmetries to reduce the number of distinct MC cases (much like Wang [2013]), but we employ a faster hybrid approach to volume computation that significantly reduces the required arithmetic operations: in many common cases where the implied polyhedron in a cell can be decomposed into pyramids and truncated prisms, we directly compute their volumes with appropriate formulas, without any boundary triangulation. In the remaining cases, similar to Wang [2013], we use the divergence theorem to compute per-cell volumes from virtually reconstructed meshes based on the MC table [Bourke 1994]. This hybrid approach minimizes unnecessary triangle reconstructions for efficiency while generating consistent results. We further distinguish our work from that of prior authors on this subject [Min and Gibou 2007; Wang 2013] by providing the full source code (in C++17) as a convenient reference for future implementers (`https://github.com/tetsuya-takahashi/MC-style-vol-eval`). For completeness, the provided code includes not only our 3D MC-style volume evaluation method, but also the corresponding 3D MC-style surface area evaluation method, along with analogous 2D MS-style area and perimeter evaluation methods (see Appendix A).

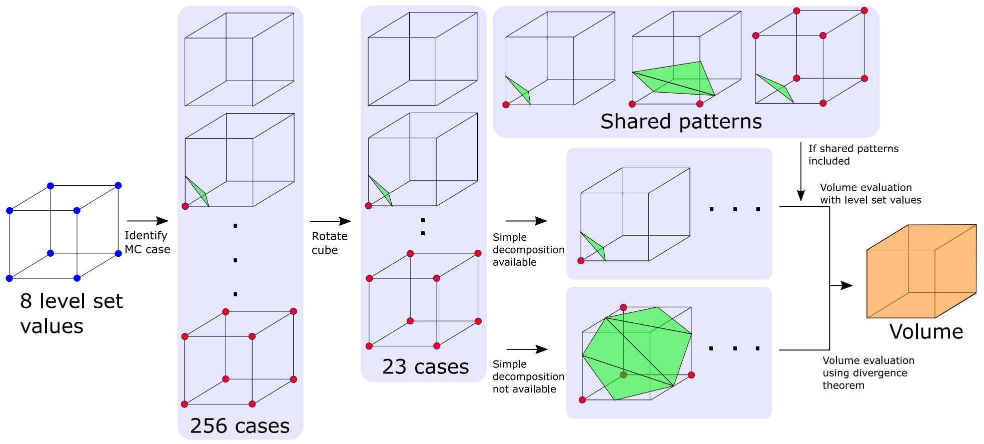
Shared patterns
8 level set values
Identify MC case
256 cases
Rotate cube
23 cases
Simple decomposition available
Simple decomposition not available
If shared patterns included
Volume evaluation with level set values
Volume
Volume evaluation using divergence theorem

Figure 2. Overview of our MC-style volume evaluation. Our method takes eight level set values for a cube and identifies its MC case. We apply rotations to the cube to match one of the predetermined 23 MC cases, and then compute the volume either from the level set values directly (exploiting shared arithmetic patterns if applicable) or from triangle areas using the divergence theorem.

2. Marching-Cubes-Style Evaluation

In this section, we describe our MC-style evaluation method for level set values ϕ defined on a uniform Cartesian grid. Following the traditional definition, we treat regions with $\phi < 0$ as inside, $\phi = 0$ as interface, and $\phi > 0$ as outside [Osher and Fedkiw 2004]. Technically, our algorithm does not rely on the grid data ϕ satisfying the eikonal equation $\|\nabla\phi\| = 1$, i.e., ϕ is not required to be a signed distance; therefore our method can be used to approximate the volume of the zero isocontour of any grid-sampled implicit surface. We process the entire space cell by cell in the MC fashion and explain how to process each cube.

2.1. Volume Evaluation

Figure 2 illustrates an overview of our volume evaluation algorithm. Our method takes eight level set values for each cube as input. Then, we identify the cube's MC case from the 256 possible cases based on the eight level set values. This operation can be performed using the publicly available MC templates [Bourke 1994].

As it is impractical to manually implement volume computations for each of the 256 cases, we utilize rotational symmetries to reduce the number of unique MC cases. Applying the 24 possible rotation patterns for a cube [Baker 2021] allows us to reduce the unique MC cases from 256 to 23 [Newman and Yi 2006] (see Figure 3 for a visualization of all the cases, numbered Case00 through Case22). Here, we note that although we have only 23 MC cases, triangulation patterns are not necessarily unique for each case, depending on the used MC table and cube rotations. Specifically, the

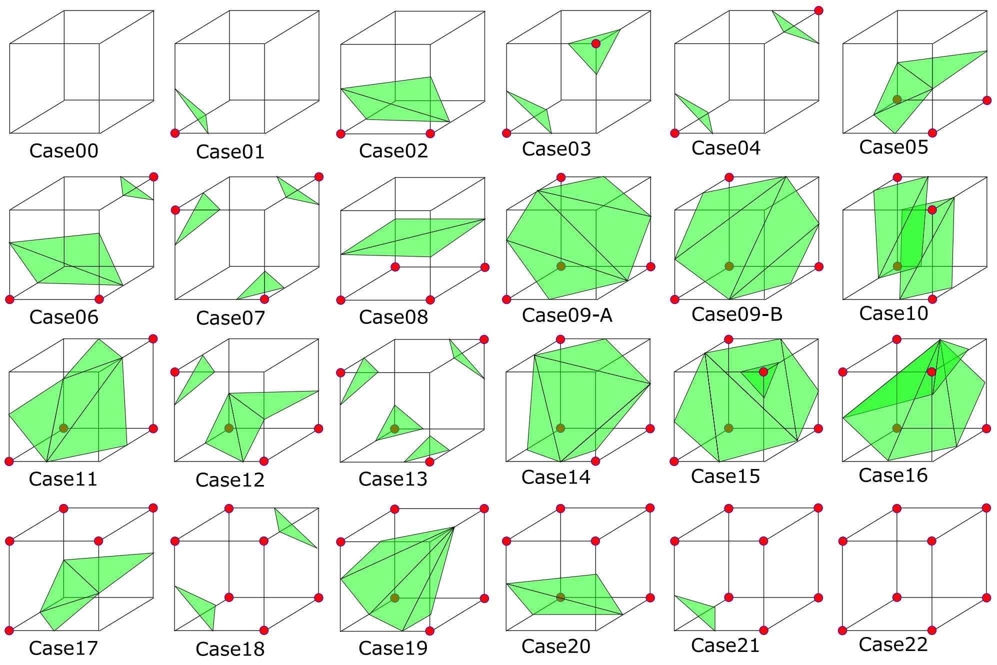

Figure 3. The 23 MC cases, with two distinct triangulation patterns for Case09 illustrated as Case09-A and Case09-B. Red disks denote inside.

widely used MC table by Bourke [1994], which we follow, needs at least two triangulation patterns for Case09 (identified as Case09-A and Case09-B in Figure 3), whereas the other cases can be uniquely triangulated with a single MC template.

The approach of Wang [2013] computed volumes for the 23 cases (albeit without distinguishing Case09-A/B) by virtually reconstructing a closed volume of triangles and applying the divergence theorem. Our key observation is that, for around two-thirds of the 23 cases, it is possible to more efficiently compute volumes directly from the level set values themselves without reconstructing triangles at all. Those cases are the ones for which the MC algorithm would generate fewer than four connected triangles. Our approach is to decompose the implied polyhedron within a cube into pyramids and truncated prisms and compute their volumes with known formulas. Although we explored the application of this approach to the remaining cases (i.e., those that yield four or more connected MC triangles), the number of decomposed shapes with newly generated boundaries becomes too large, making the direct volume computation inefficient. As such, we adopt our fast decomposition strategy where it is beneficial, and otherwise fall back to the divergence theorem–based strategy of Wang [2013] (with a few additional minor speed and consistency enhancements). Let us now consider the various cases in detail.

Case00 and Case22 are special cases that do not require any MC triangulation; we simply return 0 and 1 as their volumes, respectively.

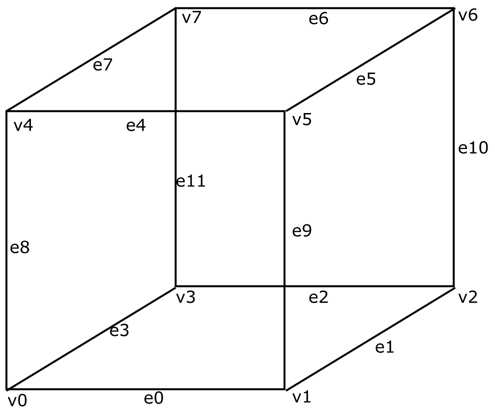

Figure 4. Definition of original vertices at the cube corners with the prefix v and newly generated vertices on the edges with the prefix e.

In the following cases, we decompose the implied polyhedron into one or more pyramids and truncated prisms whose volumes can be directly computed from level set values via the well-known volume formulas for pyramids

$$V = \frac{1}{3}Ah, \tag{1}$$

where V denotes the volume, A the area of the base polygon, and h the height of the pyramid, and truncated prisms

$$V = \frac{1}{3}A(h_1 + h_2 + h_3), \tag{2}$$

where h_1, h_2, h_3 denote the three heights of the truncated prism. These A, h, h_1, h_2, and h_3 can be directly determined from level set values. In addition, as certain decomposed shapes can appear in multiple cases, we reuse their volume computations for conciseness. We define these shared computation patterns as P1, P2, and P21 because they correspond to Case01, Case02, and Case21, respectively. (Although P1 and P21 are complementary, we found it slightly simplified the implementation to treat them distinctly.) P1, P2, and P21 take level set cube corner (vertex) values as inputs (e.g., P1(c0, c1, c3, c4) describes P1 taking the level set values at vertices 0, 1, 3, and 4). To reduce the computational cost for Case17, Case18, Case20, and Case21, we first compute the volume of the cube *outside* of the implicit surfaces and then determine the interior volume from the complement; we indicate this with the superscript *. We list the vertices of the decomposed shapes (R for pyramid and S for truncated prism), using the notation v for cube vertices and e for newly generated edge vertices (see Figure 4):

- Case01: P1(c0, c1, c3, c4)
- Case02: P2(c3, c0, c4, c2, c1, c5)

- Case03: P1(c0, c1, c3, c4), P1(c5, c1, c6, c4)
- Case04: P1(c0, c1, c3, c4), P1(c6, c7, c5, c2)
- Case05: R(v1, v2, v3, e3, e0, e9), R(v3, v2, e10, e11, e9), R(v3, e3, e9, e11)
- Case06: P2(c3, c0, c4, c2, c1, c5), P1(c6, c7, c5, c2)
- Case07: P1(c4, c0, c5, c7), P1(c1, c0, c2, c5), P1(c6, c7, c5, c2)
- Case08: S(v0, v1, v2, e8, e9, e10), S(v0, v2, v3, e8, e10, e11)
- Case10: P2(c4, c7, c6, c0, c3, c2), P2(c2, c1, c0, c6, c5, c4)
- Case12: R(v1, v2, v3, e3, e0, e11), R(v1, v2, e10, e9, e11), R(v1, e0, e9, e11), P1(c4, c0, c5, c7)
- Case13: P1(c4, c0, c5, c7), P1(c1, c0, c2, c5), P1(c6, c7, c5, c2), P1(c3, c0, c2, c7)
- Case17*: R(v1, v2, v3, e3, e0, e9), R(v3, v2, e10, e11, e9), R(v3, e3, e9, e11)
- Case18*: P21(c0, c1, c3, c4), P21(c6, c5, c2, c7)
- Case20*: R(v0, v1, e1, e3, e8), R(v1, e1, e9, e8)
- Case21*: P21(c0, c1, c3, c4)

To illustrate one nontrivial example, Figure 5 shows the decomposition into three pyramids for Case05.

For the remaining cases, we largely follow the work of Wang [2013], which computes volumes based on the divergence theorem applied to a surface triangulation. However, we improve the performance in two ways. First, to compute areas of axis-aligned faces lying on the cube's exterior, we directly compute areas of these polygons without triangulation. Second, because polygons located entirely on $x = 0$, $y = 0$, or $z = 0$ do not contribute to the volume, we simply omit these computations. In addition, to achieve perfectly consistent results with respect to Marching Cubes meshes, we address the two distinct triangulation patterns for Case09 as Case09-A and Case09-B (which Wang did not consider). In the following, we list the vertices of polygons that contribute to the volume computation, using T for triangle and Z for trapezoid (in addition to the computation pattern P1). Additionally, for pentagonal axis-aligned faces, we compute their areas as the complement of the corresponding triangle area outside the implicit surface, denoting such triangles with the superscript **.

- Case09-A: T(e10, e7, e6), T(e1, e7, e10), T(e1, e8, e7), T(e1, e0, e8), T(v2, e1, e10), T(v6, e6, e10)**, T(v7, e6, e7)

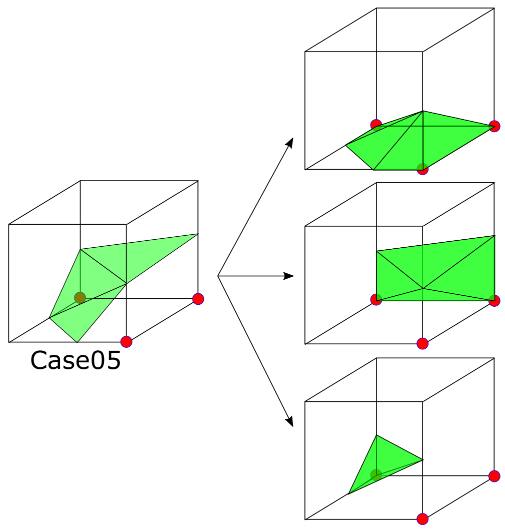

Figure 5. Example of polyhedron decomposition: the interior volume of Case05 can be determined from three pyramids.

- Case09-B: T(e10, e1, e0), T(e6, e10, e0), T(e6, e0, e8), T(e6, e8, e7), T(v2, e1, e10), T(v6, e6, e10)**, T(v7, e6, e7)
- Case11: T(e0, e8, e11), T(e0, e11, e5), T(e0, e5, e1), T(e5, e11, e6), Z(e1, v2, v6, e5), T(e6, v7, e11)**, T(e5, v6, e6)
- Case14: T(e0, e3, e7), T(e0, e7, e10), T(e0, e10, e9), T(e6, e10, e7), Z(v1, v2, e10, e9), T(e10, v6, e6)**, T(v7, e7, e6)
- Case15: T(e1, e6, e10), T(e1, e7, e6), T(e1, e0, e7), T(e8, e7, e0), T(e1, v2, e10), T(e10, v6, e6)**, T(v7, e7, e6), P1(c5, c4, c1, c6)
- Case16: T(e1, e3, e6), T(e1, e6, e10), T(e3, e8, e6), T(e5, e6, e9), T(e8, e9, e6), T(e1, v2, e10), T(e9, e5, v5), T(e10, v6, e6)**, T(e5, v6, e6)**
- Case19*: T(e8, e4, e5), T(e8, e5, e3), T(e9, e0, e5), T(e0, e3, e5), T(v5, e9, e5)**, Z(v2, v6, v7, v3), T(v5, e5, e4)**

2.2. Surface Area Evaluation

Our MC-style volume evaluation algorithm can be naturally adapted to surface area evaluation. Specifically, after we reduce the number of cases from 256 to 23 using rotational symmetries, we determine the necessary surface triangles and evaluate their areas. In this case, the difference from the work of Wang [2013] is simply that our approach uses Bourke's popular MC table [Bourke 1994], especially taking into account the two distinct triangulation patterns for Case09, to generate results perfectly consistent with the corresponding MC mesh reconstructions.

N	$\hat{v}$	ϵ_v	μ_v	$\hat{a}$	ϵ_a	μ_a
32^3	0.112376	6.37372×10^{-3}	N/A	1.12716	3.37205×10^{-3}	N/A
64^3	0.112914	1.61699×10^{-3}	3.94	1.13001	8.53825×10^{-4}	3.95
128^3	0.113052	4.03310×10^{-4}	4.01	1.13073	2.12861×10^{-4}	4.01
256^3	0.113086	1.00607×10^{-4}	4.01	1.13091	5.31158×10^{-5}	4.01
512^3	0.113094	2.51891×10^{-5}	3.99	1.13096	1.32906×10^{-5}	4.00

Table 1. Results of our MC-style evaluation with a sphere with radius $r = 0.3$ m, volume $v = \frac{4}{3}\pi r^3 = 0.113097\ \mathrm{m}^3$, and surface area $a = 4\pi r^2 = 1.13097\ \mathrm{m}^2$, where N denotes the grid resolution, $\hat{v}$ the computed volume in m^3, $\hat{a}$ the computed surface area in m^2, ϵ_v and ϵ_a relative errors for volume and area, respectively, and μ_v and μ_a the error ratios for volume and area, respectively.

3. Results and Discussions

We implemented our method in C++17. All of our experiments are executed using a single thread on a machine with an Intel Core i7-9700 CPU and 16 GB RAM.

Sphere To evaluate the correctness of our method, we first experimented with a sphere having radius r, with ground truth volume v and surface area a determined analytically by $v = \frac{4}{3}\pi r^3$ and $a = 4\pi r^2$, respectively. Table 1 shows that the computed volume and area converge toward the analytical solutions at the expected second-order rate with increasing grid resolution.

Bunny To evaluate the method in more general scenarios, we first tested with a bunny shape that has been advected for 30 frames under an incompressible velocity field generated by curl-noise [Bridson et al. 2007], with a grid resolution of 256^3 (see Figure 1 (left)). As the analytical solution is not available, we consider the ground truth to be the volume computed directly from the watertight, explicitly MC-reconstructed triangle mesh with the templates of Bourke [1994]. We implemented and compared the following four schemes:

- Supersampling, which sums up eight subvoxels' volumes per cell, determining their insideness based on the interpolated level set value at the subvoxel center (e.g., [Kim et al. 2007; Bridson 2016]);
- MT, which uses MT-style volume computation based on Bridson's implementation within Batty's viscous liquid simulator code [Batty 2013] (see `volume_fractions.cpp`), applied to six tetrahedra decomposed from a cube, with an augmentation using MC-based early pruning (if all of the cube's eight level set values are negative or positive);

- MC, which uses MC-style volume computation based on the divergence theorem approach [Wang 2013] using Bourke's popular MC template [Bourke 1994], except without proper handling of Case09-A/B;
- MC (ours).

Table 2 compares the results at the last frame. Our method is much more accurate than other methods and generated the same result as the ground truth up to machine precision; we achieve this because we designed our method to be consistent with the MC Table of Bourke [1994], taking into account the multiple triangulation patterns (Case09-A and -B), which contrasts with MC [Wang 2013]. Supersampling [Kim et al. 2007] was slower than the others because it always needs to evaluate eight level set values at the subvoxel centers. MT [Batty 2013] was slightly faster than both MC [Wang 2013] and our method in this scene because computations for individual tetrahedra within each cube can be pruned more frequently for simpler implicit surfaces. Our method is about 8% faster than Wang's, which is the next most accurate alternative. Though this net speedup is modest, note that only surface cells can benefit from our improved numerics: for shapes that have low(er) surface-to-volume ratios, the majority of cells fall into Case00 or Case22, i.e., trivially have volume 0 or 1. Depending on the application, one may be able to replace the dense grid with a narrow band data structure to near-instantly cull out most such "deep" cells, and thus the relative computational benefit of our scheme would correspondingly increase.

Random For a much more challenging scenario with a high surface-to-volume ratio, we randomly generated level set values between -1 and $+1$ with a fixed value $(+1)$ at all the outermost cells to ensure watertight meshes, using a grid resolution of 256^3 (see Figure 1 (right)). Table 2 compares the results for the four schemes. In this experiment, our result was again significantly more accurate than the others,

Scene	Scheme	$\hat{v}$	ϵ_v	T
Bunny	Supersampling [Kim et al. 2007]	0.193523	5.4300×10^{-4}	0.311
	MT [Batty 2013]	0.193608	1.0281×10^{-4}	0.119
	MC [Wang 2013]	0.193628	4.0356×10^{-7}	0.150
	MC (ours)	0.193628	2.3509×10^{-14}	0.138
Random	Supersampling [Kim et al. 2007]	0.490858	1.59518×10^{-1}	0.714
	MT [Batty 2013]	0.491359	1.60702×10^{-1}	0.923
	MC [Wang 2013]	0.423330	2.28025×10^{-6}	1.196
	MC (ours)	0.423329	2.20692×10^{-13}	0.640

Table 2. Results of volume evaluation with a deformed bunny and randomly generated level sets (Figure 1), where $\hat{v}$ denotes the computed volume in m^3, ϵ_v the relative error for volume, and T the computation time in seconds.

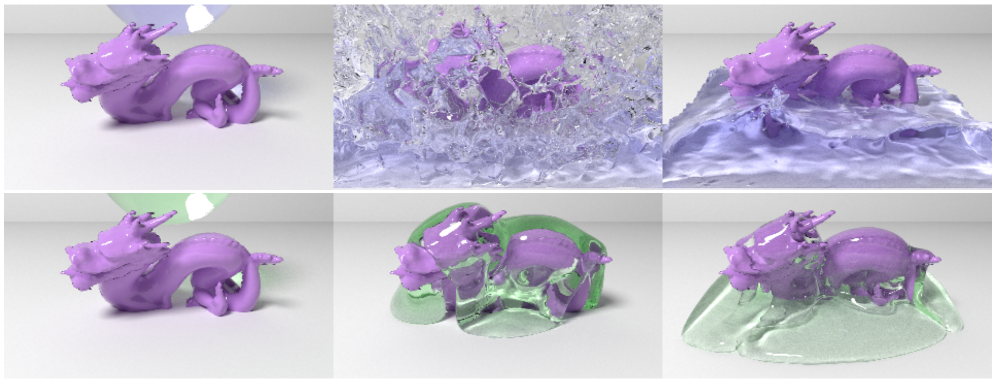

Figure 6. Inviscid (top) and viscous (bottom) liquids dropped onto a static dragon.

matching the ground truth up to machine precision. In addition, because of our direct volume evaluation and optimized area computation techniques, our method is $1.87\times$ faster than MC [Wang 2013], which may be considered a rough upper bound on our method's performance benefit. Our surface area evaluation method can also be applied to this example, and we obtained a relative error of 3.47709×10^{-13}, once again matching the ground truth value up to machine precision. Though such an apparently extreme stress test geometry will arise only infrequently in liquid animations, qualitatively similar dense foam-like (micro-)structures are increasingly important in additive manufacturing and topology optimization [Martínez et al. 2016; Zhu et al. 2017].

Spherical liquid on dragon To compare the performance of the four schemes in a practical animation scenario, we separately simulated inviscid and viscous liquid spheres dropped onto a static dragon using a grid resolution of 96^3 over 200 frames (see Figure 6) and evaluated the liquid volume at each step. Figure 7 compares the results for the computation time (left) and relative overhead with respect to our method (right), and Table 3 summarizes their averaged results. In these examples, the vast majority of cells are perfectly inside or outside and so undergo the same pruning treatment under all methods, except Supersampling [Kim et al. 2007]; thus our new hybrid scheme achieves only slightly better performance on most frames. However, when the relative liquid surface area increases, e.g., due to dramatic splashing of the inviscid liquid (for several frames starting around frame 40), our enhancements are more frequently exercised, thus exhibiting a more distinct performance benefit.

4. Conclusions

We have presented our efficient MC-style volume evaluation method, which possesses the following key features:

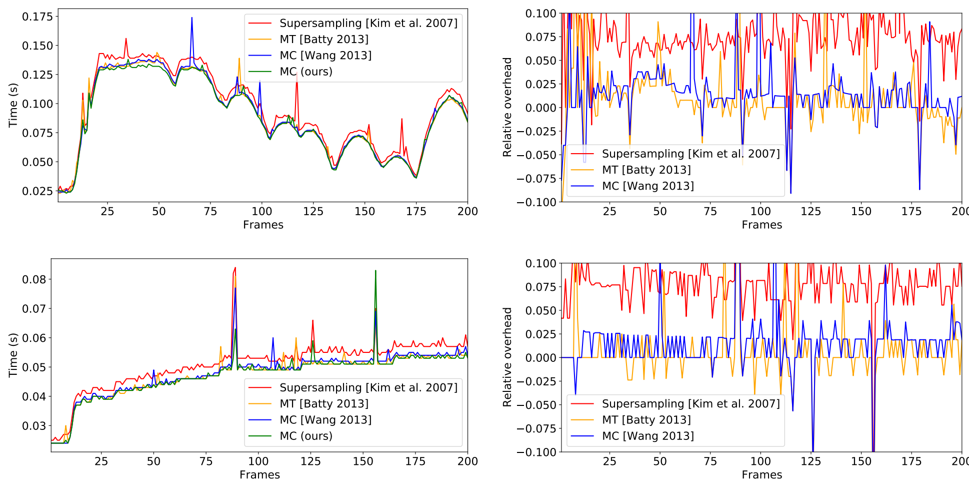

Figure 7. Profiles of the volume computation time and relative overhead with respect to our method for Figure 6 with inviscid liquids (top) and viscous liquids (bottom).

Scheme	T_i	r_i	T_v	r_v
Supersampling [Kim et al. 2007]	0.09595	0.078938	0.050905	0.080098
MT [Batty 2013]	0.08989	0.010795	0.047365	0.004986
MC [Wang 2013]	0.09050	0.017654	0.047875	0.015807
MC (ours)	0.08893	N/A	0.04713	N/A

Table 3. Performance results on volume evaluation for Figure 6, where T_i and T_v denote averaged per-frame computation times for inviscid and viscous liquids, respectively, in seconds and r_i and r_v denote the relative overhead with respect to our method for inviscid and viscous liquids, respectively.

- Our method is based on MC to avoid several MT issues (e.g., artificial bumps and directional bias), in contrast to some previous work [Min and Gibou 2007; Batty 2013], while achieving a compact implementation by exploiting rotational symmetries and reusing shared computational patterns.
- Our method correctly computes the volume in a manner precisely consistent with a widely used set of MC templates [Bourke 1994], unlike prior work [Wang 2013; Min and Gibou 2007; Kim et al. 2007; Batty 2013], and demonstrably converges at a second-order rate;
- Our method directly computes the volume from level set values in approximately two-thirds of the MC cases for efficiency while relying on the divergence-theorem for the remaining cases, achieving faster performance over the work of Wang [2013].

Although we focused on consistency with respect to one specific and commonly used set of MC templates [Bourke 1994], our overall strategy should be naturally extensible to other tables and variants (e.g., Marching Cubes 33 [Chernyaev 1995]).

Our C++17 implementation of MC-style volume and surface area evaluation and MS-style area and perimeter evaluation is available online at `https://github.com/tetsuya-takahashi/MC-style-vol-eval`. Just as the many existing freely available codes/tables for Marching Cubes serve as convenient workhorse routines for surface reconstruction, we hope that our tool can play a similar role for level set volume, area, and length evaluations in both two and three dimensions.

A. Marching-Squares-Style Evaluation

Our framework can also be adapted to 2D MS-style evaluations of area and perimeter for implicit lines, which is useful for 2D problems as well as slices of 3D problems (e.g., for determining partial face areas for cut-cell finite volume fluid codes). However, unlike MC, MS takes four level set values and has only 16 $(= 2^4)$ cases. Given the comparatively small number of MS cases, we directly compute the area and perimeter of the implicit lines without utilizing rotational symmetries except for the well-known ambiguous cases, whose implementation is slightly more involved. We disambiguate these cases by checking whether the level set value at the cell center (i.e., the average of the four corner values) is less than zero.

Circle Analogous to our 3D sphere test, we evaluated the correctness of our 2D MS-style area and perimeter evaluation with a circle (radius $r = 0.3$ m), whose area a and perimeter p can be analytically computed by $a = \pi r^2$ and $p = 2\pi r$, respectively. Table 4 shows that the computed areas and perimeters converge toward the analytical solution at the expected second-order rate with increasing grid resolution.

N	$\hat{a}$	ϵ_a	μ_a	$\hat{p}$	ϵ_p	μ_p
32^2	0.282138	2.14238×10^{-3}	N/A	1.88370	6.65718×10^{-4}	N/A
64^2	0.282579	5.81890×10^{-4}	3.68	1.88464	1.65452×10^{-4}	4.02
128^2	0.282704	1.40595×10^{-4}	4.14	1.88488	4.14497×10^{-5}	3.99
256^2	0.282734	3.44212×10^{-5}	4.08	1.88494	1.03709×10^{-5}	4.00
512^2	0.282741	8.80409×10^{-6}	3.91	1.88495	2.59050×10^{-6}	4.00

Table 4. Results of MS-style area and perimeter evaluations with a circle with radius $r = 0.3$ m, area $a = \pi r^2 = 0.282743\ \mathrm{m}^2$, and perimeter $p = 2\pi r = 1.88496$ m, where N denotes the grid resolution, $\hat{a}$ the computed area in m^2, $\hat{p}$ the computed perimeter in m, ϵ_a and ϵ_p the relative errors for area and perimeter, respectively, and μ_a and μ_p the error ratios for area and perimeter, respectively.

Random Lastly, we perform the 2D equivalent of our 3D random stress test, again using randomly generated level set values (between -1 and $+1$) with a fixed positive value ($+1$) on the outermost cells to ensure watertightness. In this experiment, the relative errors of our method for area and perimeter are 2.68864×10^{-15} and 1.91267×10^{-14}, respectively.

Index of Supplemental Materials

The single header-only C++17 source code of our method (which relies on the header-only library Eigen (3.4.0) [Guennebaud et al. 2010] for simple 3D vector operations, such as dot and cross products) for our MC-style volume and surface area evaluation and MS-style area and perimeter evaluation is provided, along with a `main.cpp` file to demonstrate its usage (`https://github.com/tetsuya-takahashi/MC-style-vol-eval`). This code has been successfully built and run on Ubuntu 20.04.1 LTS with g++ 9.3.0 and Windows 10 with Visual Studio 2019.

References

BAKER, M. J., 2021. Maths—Cube rotation. URL: `https://www.euclideanspace.com/maths/discrete/groups/categorise/finite/cube/index.htm`. 33

BATTY, C., 2013. Variationalviscosity3d. URL: `https://github.com/christopherbatty/VariationalViscosity3D`. 31, 38, 39, 41

BOURKE, P., 1994. Polygonising a Scalar Field. URL: `http://paulbourke.net/geometry/polygonise/`. 32, 33, 34, 37, 38, 39, 41, 42

BRIDSON, R., HOURIHAN, J., AND NORDENSTAM, M. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics 26*, 3 (July), 46–es. URL: `https://doi.org/10.1145/1276377.1276435`, doi:10.1145/1276377.1276435. 38

BRIDSON, R. 2016. *Fluid Simulation for Computer Graphics*, 2 ed. A K Peters/CRC Press, Boca Raton, FL. URL: `https://www.routledge.com/Fluid-Simulation-for-Computer-Graphics/Bridson/p/book/9781482232837`. 31, 32, 38

CHERNYAEV, E. V., 1995. Marching Cubes 33: Construction of topologically correct isosurfaces. Presented at GRAPHICON '95, Saint Petersburg, Russia, July 3–7, 1995, Institute for High Energy Physics, Moscow, Russia, Report CN/95-17. 42

DOI, A., AND KOIDE, A. 1991. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE Transactions on Information and Systems 74*, 214–224. URL: `https://search.ieice.org/bin/summary.php?id=e74-d_1_214`. 31

GUENNEBAUD, G., JACOB, B., ET AL., 2010. Eigen v3. URL: `http://eigen.tuxfamily.org`. 43

HANSEN, C. D., AND JOHNSON, C. R., Eds. 2004. *The Visualization Handbook*. Elsevier, Amsterdam. URL: https://www.elsevier.com/books/visualization-handbook/hansen/978-0-12-387582-2. 31

KIM, B., LIU, Y., LLAMAS, I., JIAO, X., AND ROSSIGNAC, J. 2007. Simulation of bubbles in foam with the volume control method. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, Association for Computing Machinery, 98–es. URL: https://doi.org/10.1145/1275808.1276500, doi:10.1145/1275808.1276500. 31, 38, 39, 40, 41

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching Cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics 21*, 4 (Aug.), 163–169. URL: https://doi.org/10.1145/37402.37422, doi:10.1145/37402.37422. 31

MAPLE, C. 2003. Geometric design and space planning using the marching squares and marching cube algorithms. In *2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings*, IEEE, 90–95. URL: https://ieeexplore.ieee.org/document/1219671, doi:10.1109/GMAG.2003.1219671. 31

MARTÍNEZ, J., DUMAS, J., AND LEFEBVRE, S. 2016. Procedural voronoi foams for additive manufacturing. *ACM Transactions on Graphics 35*, 4, 44:1–44:12. URL: https://doi.org/10.1145/2897824.2925922, doi:10.1145/2897824.2925922. 40

MIN, C., AND GIBOU, F. 2007. Geometric integration over irregular domains with application to level-set methods. *Journal of Computational Physics 226*, 2, 1432–1443. URL: https://www.sciencedirect.com/science/article/pii/S0021999107002410, doi:10.1016/j.jcp.2007.05.032. 31, 32, 41

NEWMAN, T. S., AND YI, H. 2006. A survey of the marching cubes algorithm. *Computers & Graphics 30*, 5, 854–879. URL: https://www.sciencedirect.com/science/article/pii/S0097849306001336, doi:10.1016/j.cag.2006.07.021. 33

OSHER, S., AND FEDKIW, R. 2004. *The Level Set Methods and Dynamic Implicit Surfaces*, vol. 153 of *Applied Mathematical Sciences*. Springer, New York. URL: https://link.springer.com/book/10.1007/b98879, doi:10.1115/1.1760520. 31, 33

WANG, S., 2013. 3D volume calculation for the marching cubes algorithm in Cartesian coordinates. URL: https://arxiv.org/abs/1308.0387, arXiv:1308.0387. 32, 34, 36, 37, 39, 40, 41

ZHU, B., SKOURAS, M., CHEN, D., AND MATUSIK, W. 2017. Two-scale topology optimization with microstructures. *ACM Transactions on Graphics 36*, 4, 120b:1–120b:16. URL: https://doi.org/10.1145/3072959.3095815, doi:10.1145/3072959.3095815. 40

Author Contact Information

Tetsuya Takahashi
Adobe Inc.
345 Park Ave
San Jose, CA 95110
ttakahas@adobe.com
https://tetsuya-takahashi.github.io/

Christopher Batty
University of Waterloo
200 University Ave W
Waterloo, ON N2L 3G1, Canada
christopher.batty@uwaterloo.ca
https://cs.uwaterloo.ca/~c2batty/

Tetsuya Takahashi and Christopher Batty, Fast Marching-Cubes-Style Volume Evaluation for Level Set Surfaces, *Journal of Computer Graphics Techniques (JCGT)*, vol. 11, no. 2, 30–45, 2022
http://jcgt.org/published/0011/02/02/

Received:	2021-11-01		
Recommended:	2022-01-29	Corresponding Editor:	Christophe Hery
Published:	2022-06-30	Editor-in-Chief:	Marc Olano