Resolving Non-Manifoldness on Meshes from Dual Marching Cubes

D. Zint[1,2] and R. Grosso[1] and P. Gürtler[1]

[1] Visual Computing, Friedrich-Alexander-University Erlangen-Nuremberg, Germany
[2] TITANE Project-Team, INRIA Sophia Antipolis Méditerrannée, France

Abstract

There are several methods that reconstruct surfaces from volume data by generating triangle or quad meshes on the dual of the uniform grid. Those methods often provide meshes with better quality than the famous marching cubes. However, they have a common issue: the meshes are not guaranteed to be manifold. We address this issue by presenting a post-processing routine that resolves all non-manifold edges with local refinement. New vertices are positioned on the trilinear interpolant. We verify our method on a wide range of data sets and show that we are capable of resolving all non-manifold issues.

CCS Concepts

• ***Computing methodologies*** → *Shape analysis; Mesh geometry models; Parallel algorithms;*

1. Introduction

Surface reconstruction from volume data is a well covered problem in computer graphics and medical imaging. The sampled data is often approximated with the trilinear interpolant,

$$\begin{aligned} T(u,v,w) = \\ &(1-w)[f_0(1-u)(1-v)+f_1u(1-v) \\ &+f_2(1-u)v+f_3uv] \\ &+w[f_4(1-u)(1-v)+f_5u(1-v) \\ &+f_6(1-u)v+f_7uv], \end{aligned} \tag{1}$$

where $(u,v,w) \in [0,1)^3$ are the local coordinates within a cell of the volume data. The scalar value at the cell vertex v_i is denoted by f_i. The reconstructed surface is determined by the iso-value ι_0,

$$S_{\iota_0} := \{(u,v,w) \,|\, T(u,v,w) = \iota_0\}. \tag{2}$$

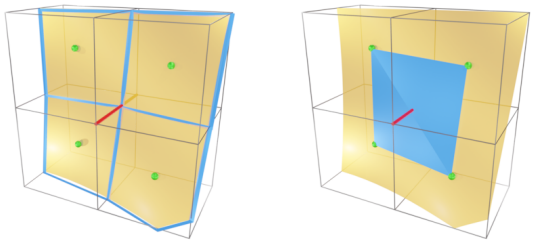
The trilinear interpolant is continuous across cell borders, i.e. the reconstructed surface is C^0-continuous.

Marching Cubes (MC) [LC87] is probably the most famous surface-reconstruction method. It approximates the surface with triangles in each cell of the volume data independently and therefore it can be easily parallelized. However, meshes generated with MC have some drawbacks. They usually contain poorly shaped triangles, i.e. small angles, and are not watertight, due to inconsistencies across cell borders. Several methods were presented in the last decades that extended MC. The issue with non-watertight meshes was fixed with the asymptotic decider that resolves ambiguous cases at cell borders [NH91, LLVT03]. Furthermore, within a cell, MC does not represent the topology of the iso-surface correctly. There are several methods that aim for topological correctness [Gro16, Gro17, RWY05]. Nevertheless, the issue with poorly shaped triangles remains. Other methods specifically constructed for watertight and 2-manifold surfaces produce very good results but are hard to efficiently parallelize due to their dependence on octrees or the generation of tetrahedral meshes [HZG20, TPG99].

Starting with the Surface Nets method [Gib98], alternatives to the classical MC came up that consider the dual of the marching cubes surfaces. While MC generates triangles for the iso-surface within a cell, dual methods approximate the iso-surface with vertices that are then connected to elements across cell borders. If the volume data is stored in a voxel grid, then the resulting mesh will be quad only. This is advantageous as quad meshes require less memory than triangle meshes but even when the quads are split into triangles, e.g. for visualization, the resulting meshes are of better quality. A comparison between MC and Surface Nets was given by de Bruin et al. [dBVP*00]. The idea of Surface Nets was extended by Nielson who brought up the name Dual Marching Cubes (DMC) [Nie04]. This method produces manifold meshes but it does not follow the topology of the trilinear interpolant as the method relies on a look-up table. A variation of this method suitable for parallel implementation was presented in [LS12].

The method Dual Contouring [JLSW02] also produces one vertex for each cube that contains an iso-surface and has therefore some similarities with Surface Nets. The vertex placement has the objective of representing sharp edges correctly by computing the vertex positions with a quadric error function. A major difference to Surface Nets is that Dual Contouring was designed for octrees. Several methods were built on the idea of Dual Contouring [ZHK04, SW04, SJW07, KKDH07, RSA16]. Octrees have the advantage that manifoldness can be always achieved by refinement. However, implementing octrees efficiently is challenging, es-

DOI: 10.2312/egs.20221029



(a) voxel edge with its four adjacent cells (b) quad around voxel edge

Figure 1: Generation of quads in the DMC method.

pecially on GPUs. Additionally, the methods loose the ability to produce quad meshes, when using octrees.

Another Dual Marching Cubes method was presented in [GZ21]. This method considers the fact that the iso-surface can have several independent branches within one cell. Each branch is represented by its own vertex. This solves some of the topological issues of Dual Contouring. For example, non-manifold vertices can only appear at boundaries. Additionally, it is rather simple to implement and parallelize this method as it works with voxel grids instead of octrees.

In this work, we discuss the appearance of non-manifold edges in the DMC method from [GZ21] and present a post-processing step to remove them. We also show how to remove non-manifold vertices at boundaries with a simple pre-processing step. With this two extensions we produced manifold meshes in all our test cases. Our method was implemented in CUDA and is available on GitHub[†].

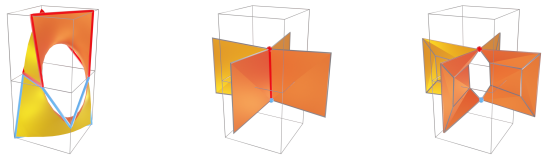
2. Dual Marching Cubes

Our post-processing step for resolving non-manifold edges is applicable to any of the DMC methods mentioned in Section 1. However, we restrict ourselves to the method presented in [GZ21]. The advantage of this DMC method is that it does not produce any non-manifold vertices on the interior of the volume data. Thus, we only need to consider non-manifold edges. From here on, whenever we use the term DMC, we refer to this method.

The edges in a voxel grid always have four adjacent cells. DMC generates a quad for each voxel edge that is intersected by the iso-surface, Figure 1a. The iso-surface within each cell is represented by a vertex. The quad is formed by connecting the vertices of the four cells, Figure 1b. Thus, the resulting mesh will be quad only.

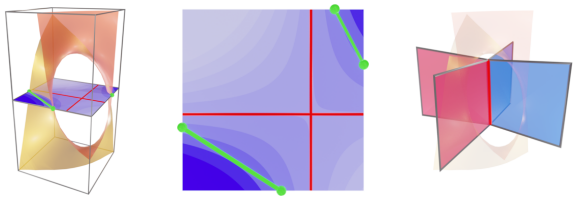
Non-manifold edges appear, when the iso-surfaces of two neighboring cells form a tunnel, Figure 2a. In that case, quads are generated for all four edges of the face in between the two cells and the tunnel is replaced by a non-manifold edge, Figure 2b. At the boundary of the voxel grid, non-manifold edges with only three incident quads can appear because the fourth quad would protrude the grid. Also non-manifold vertices near boundaries are caused by that. More details on such boundary cases are given in Section 4.

† https://github.com/PhiliGuertler/ManifoldDMC



(a) tunnel (b) non-manifold edge (c) resolved edge

Figure 2: DMC produces a non-manifold edge if the iso-surface forms a tunnel in between two cells. We resolve the non-manifoldness by restoring the tunnel of the iso-surface.



(a) tunnel (b) asymptotic decider (c) correct topology

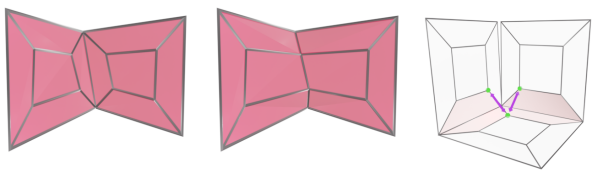
Figure 3: We use the asymptotic decider on the face in between the two cells for finding the orientation of the iso-surface tunnel.

3. Non-Manifold Edges

Non-manifold edges (NMEs) are resolved by restoring the tunnel present in the iso-surface, Figure 2. First, the correct orientation of the iso-surface tunnel is determined with the asymptotic decider, Figures 3a and 3b. Thus, we get two face-pairs, as every NME consists of four incident faces, Figure 3c. Next, all faces are subdivided, Figure 4a. The faces incident to the NME are removed and vertices are merged according to the face-pairs, Figure 4b.

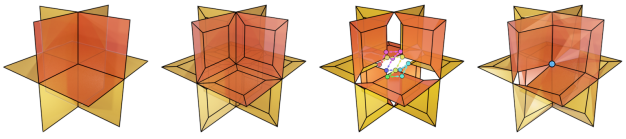
The procedure of subdivision, face removal, and vertex merging is straight forward as long as there are no other NMEs interfering. However, NMEs can be adjacent to each other. In such a case, several vertices have to be merged into one, Figure 4c. Therefore, we need to store all vertices that need to be merged in a list. Up to twelve vertices can be merged into one. An example for that is given in Figure 5 where a single vertex is incident to 6 NMEs.

In some rare cases, a NME is created in the merging step. A configuration consisting of five adjacent NMEs is given in Figure 6. The NME that is colored in light blue in the figure appears again



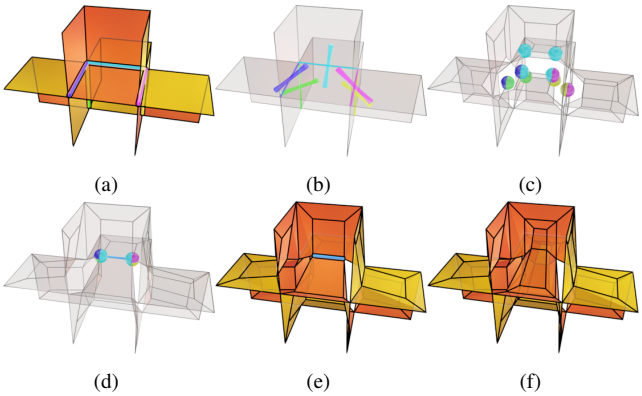
(a) subdivision (b) merge (c) adjacent NMEs

Figure 4: Geometric manifoldness is achieved by subdividing the quads and merging the nodes that surround the former non-manifold edge.



(a) NMEs (b) subdivision (c) face removal (d) merge

Figure 5: A case where six NMEs are incident to a single vertex.



(a) (b) (c) (d) (e) (f)

Figure 6: Sometimes, not all NMEs can be resolved at once. We run our method iteratively to ensure that no more NMEs are left.

after the merging step, Figure 6d. The process is repeated iteratively until all NMEs are removed. In our tests, we never needed more than two iterations for removing all NMEs.

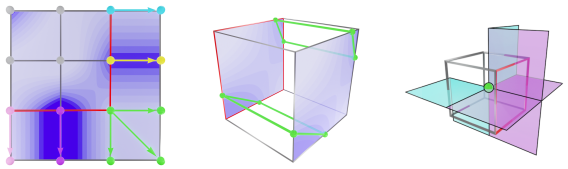
4. Non-Manifoldness on Boundaries

DMC generates quads across voxel edges. Therefore, boundaries cause some special cases that include non-manifold vertices, Figure 7c, and NMEs with only three incident edges. When the iso-surface intersects with a voxel edge at the boundary, DMC does not generate a quad as it would on the interior. Thus, NMEs in cells at the boundary are missing one quad. The fact that DMC does not generate quads at boundaries also causes non-manifold vertices.

All boundary issues can be resolved by adding another layer of cells around the voxel grid and copying the scalar values from the old to the new boundary, Figure 7a. A requirement for NMEs is that one face is intersected twice by the same iso-surface branch. Duplicating the boundary scalar values prohibits this case in the new boundary cells, Figure 7b. NMEs that had only three incident faces now have four. Irregular vertices disappear completely because their one ring is closed. Thus, only NMEs with four incident edges remain and they can be resolved with the procedure described in Section 3.

5. Results

We tested our method on more than twenty qualitatively different data sets, all presented in [GZ21]. Here, we restrict ourselves to the ten most interesting results, Table 1. They are visualized in Figure 8. The data sets vary significantly in the number of NMEs.



(a) (b) (c)

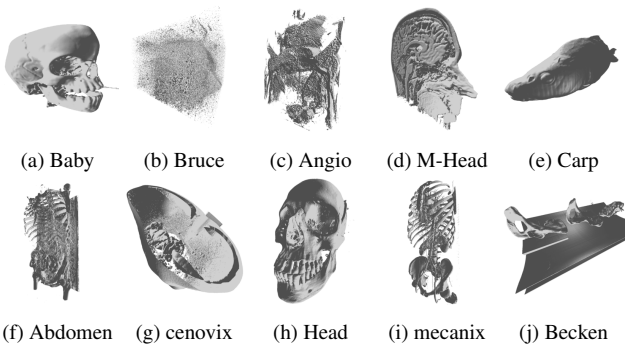
Figure 7: (a),(b) Extending the grid. Red edges correspond to the old boundaries. (c) A non-manifold vertex. Purple quads are not generated as they are partially outside the domain.

Data Set	ι_0	Size	N4	N3	NV
Baby	147.5	$(256^2, 98)$	113	0	1
Bruce	147.5	$(256^2, 156)$	37,672	593	1,121
Angio	47	$(384, 512, 80)$	1,066	73	257
M-Head	105	(256^3)	3,618	5	13
Carp	1,500	$(256^2, 512)$	13	0	0
Abdomen	80	$(512^2, 147)$	23,762	35	116
cenovix	900	$(512^2, 361)$	11,721	68	163
head	900	$(512^2, 641)$	22,275	0	0
mecanix	1,200	$(512^2, 743)$	8,487	3	22
becken	900	$(512^2, 1,047)$	36	0	0

Table 1: Data sets for evaluating our method for removing non-manifoldness. The last three columns show the number of non-manifold edges with four incident quads (N4), three incident quads (N3), and the number of non-manifold vertices (NV).

The additional layer of cells with the duplicated boundary values increase the memory consumption slightly. The relatively largest increase was measured for the data set *Baby* from 25.69 MB to 26.63 MB which is 3.64 %. The worst runtime for resolving NMEs was measured on mesh *head* and was 668.5ms. The whole generation process took 754.6ms. All tests were performed on an Nvidia Geforce GTX 1660ti.

All meshes from Table 1 were generated with DMC and contain NMEs with four incident faces, Table 1. Most of them also contain non-manifold vertices and NMEs with three incident faces. Extending the grid removed all non-manifold vertices and NMEs



(a) Baby (b) Bruce (c) Angio (d) M-Head (e) Carp

(f) Abdomen (g) cenovix (h) Head (i) mecanix (j) Becken

Figure 8: Meshes generated with DMC for the data sets in Table 1.

with three incident faces in all tests. Latest after two iterations of the NME removal all meshes from our experiments were manifold.

We show correctness of the mesh topology after post-processing by comparing them to the results of DMC on refined voxel grids, where the scalar values at new voxel vertices were computed by trilinear interpolation. Figures 9 to 12 show first the non-manifold edges, then the solution of post-processing, and finally the refined grid. All examples show geometry from our test data sets, Table 1. In all cases, the topology of the refined and the post-processed meshes is identical.

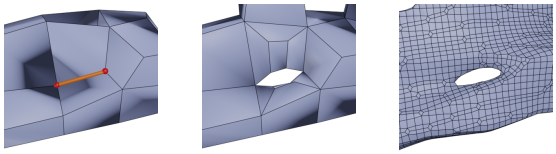


Figure 9: Resolving a NME in comparison with the refined grid.

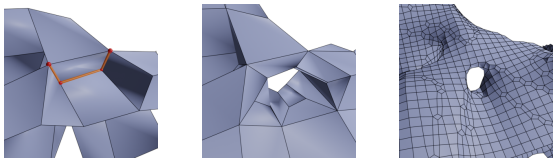


Figure 10: Resolving a quad with three NMEs in comparison with the refined grid.

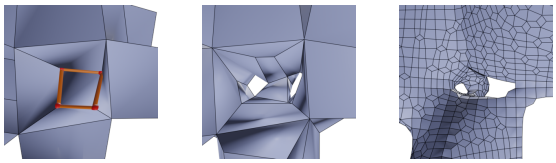


Figure 11: Resolving a quad with four NMEs in comparison with the refined grid.

6. Conclusions

We presented a method for resolving non-manifoldness on grids generated with dual marching cubes methods. Non-manifold edges and vertices were resolved in all our tests and topology was reconstructed correctly according to the iso-surface.

References

[dBVP*00] DE BRUIN P. W., VOS F. M., POST F. H., FRISKEN-GIBSON S. F., VOSSEPOEL A. M.: Improving Triangle Mesh Quality with SurfaceNets. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2000* (2000), Lecture Notes in Computer Science, Springer, pp. 804–813. 1

[Gib98] GIBSON S. F. F.: Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98* (1998), Lecture Notes in Computer Science, Springer, pp. 888–898. 1

[Gro16] GROSSO R.: Construction of Topologically Correct and Manifold Isosurfaces. *Computer Graphics Forum 35*, 5 (2016), 187–196. 1

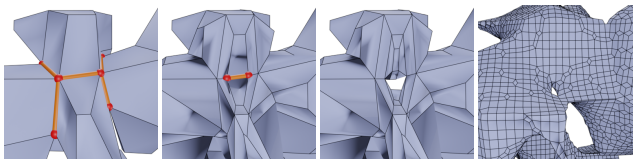


Figure 12: Resolving five NMEs iteratively in comparison with the refined grid.

[Gro17] GROSSO R.: An asymptotic decider for robust and topologically correct triangulation of isosurfaces: topologically correct isosurfaces. In *Proceedings of the Computer Graphics International Conference* (2017), CGI '17, Association for Computing Machinery, pp. 1–5. 1

[GZ21] GROSSO R., ZINT D.: A parallel dual marching cubes approach to quad only surface reconstruction. *The Visual Computer* (2021), 1–16. 2, 3

[HZG20] HUANG J., ZHOU Y., GUIBAS L.: Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621* (2020). 1

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. *ACM Trans. Graph. 21*, 3 (2002), 339–346. 1

[KKDH07] KAZHDAN M., KLEIN A., DALAL K., HOPPE H.: Unconstrained isosurface extraction on arbitrary octrees. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (2007), SGP '07, Eurographics Association, p. 125–133. 1

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics 21*, 4 (1987), 163–169. 1

[LLVT03] LEWINER T., LOPES H., VIEIRA A. W., TAVARES G.: Efficient Implementation of Marching Cubes' Cases with Topological Guarantees. *Journal of Graphics Tools 8*, 2 (2003), 1–15. 1

[LS12] LÖFFLER F., SCHUMANN H.: Generating Smooth High-Quality Isosurfaces for Interactive Modeling and Visualization of Complex Terrains. In *Vision, Modeling and Visualization* (2012), The Eurographics Association. 1

[NH91] NIELSON G. M., HAMANN B.: The asymptotic decider: Resolving the ambiguity in marching cubes. In *Proceeding Visualization '91* (1991), IEEE Computer Society Press, p. 83–91. 1

[Nie04] NIELSON G.: Dual marching cubes. In *IEEE Visualization 2004* (2004), pp. 489–496. 1

[RSA16] RASHID T., SULTANA S., AUDETTE M. A.: Watertight and 2-manifold Surface Meshes Using Dual Contouring with Tetrahedral Decomposition of Grid Cubes. *Procedia Engineering 163* (2016), 136–148. 1

[RWY05] RENBO X., WEIJUN L., YUECHAO W.: A Robust and Topological Correct Marching Cube Algorithm Without Look-Up Table. In *The Fifth International Conference on Computer and Information Technology (CIT'05)* (2005), pp. 565–569. 1

[SJW07] SCHAEFER S., JU T., WARREN J.: Manifold dual contouring. *IEEE Transactions on Visualization & Computer Graphics 13*, 03 (2007), 610–619. 1

[SW04] SCHAEFER S., WARREN J.: Dual marching cubes: primal contouring of dual grids. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.* (2004), pp. 70–76. 1

[TPG99] TREECE G. M., PRAGER R. W., GEE A. H.: Regularised marching tetrahedra: improved iso-surface extraction. *Computers & Graphics 23*, 4 (1999), 583–598. 1

[ZHK04] ZHANG N., HONG W., KAUFMAN A.: Dual contouring with topology-preserving simplification using enhanced cell representation. In *IEEE Visualization 2004* (2004), pp. 505–512. 1