Supplemental Material for:
VoroMesh: Learning Watertight Surface Meshes with Voronoi Diagrams
ICCV '23

Nissim Maruani
Inria, Université Côte d'Azur
nissim.maruani@inria.fr

Roman Klokov
LIX, École Polytechnique, IP Paris
klokov@lix.polytechnique.fr

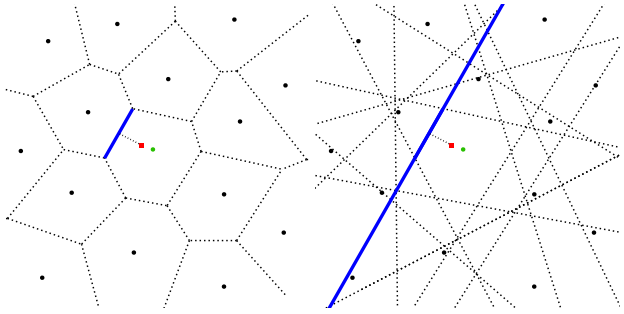
Maks Ovsjanikov
LIX, École Polytechnique, IP Paris
maks@lix.polytechnique.fr

Pierre Alliez
Inria, Université Côte d'Azur
pierre.alliez@inria.fr

Mathieu Desbrun
Inria Saclay - Ecole Polytechnique
mathieu.desbrun@inria.fr

In Section 1, we provide a proof of Theorem 1 from our ICCV paper and discuss how our VoroLoss can be implemented efficiently using a k-nearest-neighbors algorithm. We then present a detailed analysis of the chamfer distances obtained with our model trained on the ABC dataset in Section 2, before demonstrating additional qualitative comparisons for both optimization-based and learning-based results in Section 3. Section 4 discusses how to improve the outputs of VoroMesh, while a comparison of our approach to DMT [7] is presented in Section 5. Finally, additional timings of our experiments are provided in Section 6.

1. Additional comments about VoroLoss

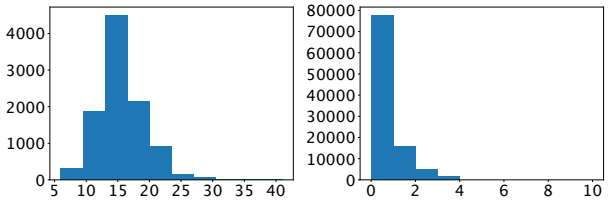


(a) Closest face (b) Closest bisector

Figure 1: Visual proof of our *VoroLoss*: given a point (red) of the target surface, the distance to the closest face in the Voronoi diagram (a) is equal to the distance to the closest bisector between the generator (green, in which the sampled point lies) and all the other ones (b).

Proof of Theorem 1. To validate our Voroloss, we need to prove that the distance from a point $x \in \mathbb{R}^3$ to the set of cell faces of the Voronoi diagram of the generators equates the distance from x to the bisector planes between the generator q_i of the Voronoi cell containing x and the other generators $q_{j\neq i}$. Note that the Voronoi cell V_i of a generator q_i is the intersection of half-spaces containing q_i [1] (see Figure 1), i.e., $V_i = \cap_{j\neq i} H^i_{i,j}$ where $H^i_{i,j} = \{y \in \mathbb{R}^3 | \|y - q_i\| \leq \|y - q_j\|\}$ denotes the half-space of points closer to q_i than q_j, where the boundary $\partial H^i_{i,j}$ of $H^i_{i,j}$ is the bisector plane $H_{i,j}$ between q_i and q_j. Therefore, each Voronoi cell is convex, implying the property we stated since:

$$d(x, \partial V_i) = d(x, \cup_{j\neq i} H_{i,j}) = \min_{j\neq i} d(x, H_{i,j}).$$



(a) #neighbors per Voronoi cell (b) Index of closest bisector

Figure 2: Statistics for a random 3D Voronoi diagram.

VoroLoss implementation details. Our *VoroLoss* can be efficiently computed using a k-nearest-neighbors algorithm. In this experiment, we consider the Voronoi diagram of 10^5 points placed randomly in $[-1, 1]^3$. Let i be the index of the closest generator for a point x of the dense sampling. The average number of faces for each Voronoi cell (which also corresponds to the degree of each vertex in the dual (Delaunay) triangulation) can be large, see Figure 2a. However, when sorting the generators by distance to the sampled point, the j-th index of the generator for which $d(x, H_{i,j})$ is minimal is low, see Figure 2b. Note that we include an implementation of *VoroLoss* (with the complete architectures of our networks) in the file `voromesh.py` of our supplemental material.

2. Quantitative Analysis

In order to complement our ICCV '23 paper, we provide histograms of chamfer distance (in logarithmic values to ex-

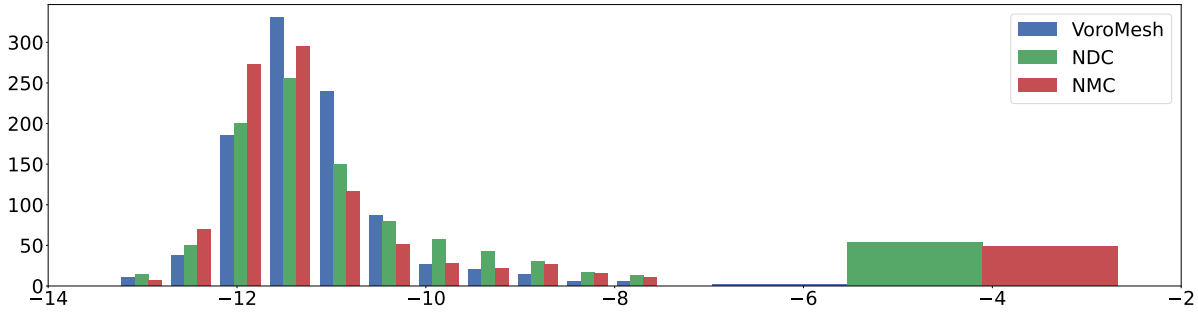


Figure 3: Histogram of Chamfer distance log values for VoroMesh, NMC [3], and NDC [2] for SDF inputs of resolution 32^3.

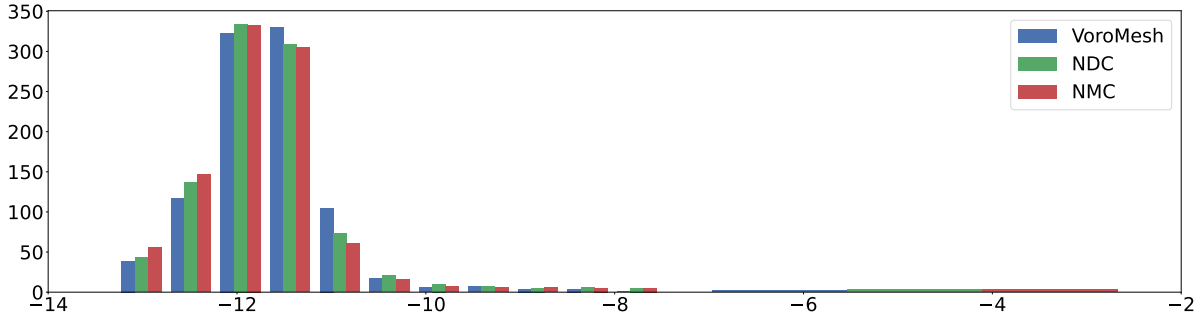


Figure 4: Histogram of Chamfer distance log values for VoroMesh, NMC [3], and NDC [2] for SDF inputs of resolution 64^3.

acerbate differences) obtained on the ABC dataset with our *VoroMesh* for input SDF grids of resolutions 32^3 and 64^3, and compare it to the state of the art in Figures 3-4.

Generally, *VoroMesh* produces slightly fewer low-CD reconstructions compared to NMC [3] and NDC [2], but it also produces significantly fewer failed reconstructions (i.e., with outlier high-CD values), which allows our method to achieve better aggregate metric values. Our approach is most effective at low resolution, where the previous observation is most obvious; additionally, *VoroMesh* produces significantly more reconstructions in the average quality range. For higher resolution, *VoroMesh* remains competitive, showing that our loss can be effectively used to produce training signal for detailed high-quality reconstructions. For a grid size 64^3, VoroMesh still outperforms competitors in aggregate CD values due to its ability to represent finer shapes with adaptive surface discretization and, as a result, its lack of failed reconstructions.

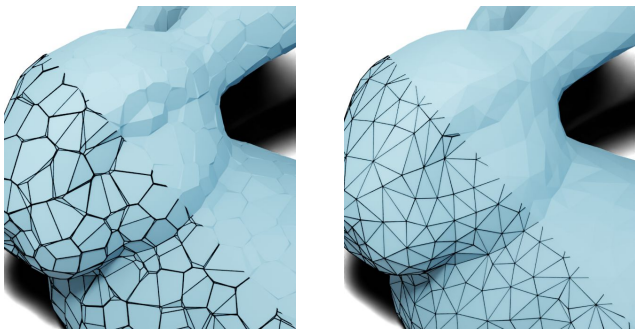
3. Qualitative Analysis

Finally, we present additional renders of the models presented in the articles for all resolutions. We also showcase the different methods on additional models.

The optimization-based results are presented in Figures 8, 9, 10, 11, and 12. Our representation is accurate and efficiently captures small details, visually outperforming all methods.

The learning-based results on the test set of ABC are presented in Figures 13, 14, and 15, while Thingi32 shapes are in Figures 16, 17. Our approach captures thin structures, but sometimes fails to correctly predict the occupancy of some cells, resulting in shadowed voids.

Note finally that the small faces visible on close inspection could be eliminated in a post-processing stage as mentioned as future work in the paper and discussed in the next section — see also Figures 5a-5b.



(a) Voromesh (b) Simplified dual Voromesh

Figure 5: A possible approach to eliminating artifacts caused by small facets (a) is to dualize the Voromesh, and collapse small edges (b).

4. Surface Artifacts

Voronoi generators in near co-circular positions can create small facets (visible in Figure 5a) which can cause shad-

ing artifacts. Removing them would improve visual quality and performance in F1 and NC metrics. In this paper, we favored strong topological guarantees over appearance; nonetheless, we explored ways to improve our reconstructions through postprocessing. One possibility is to leverage the fact that the *surface dual* to our *VoroMesh* is a triangle mesh, which can be simplified through edge collapses to yield an artifact-free surface with better shaped triangles (see Figure 5b), potentially at the expense of some details. Alternatively, our method is capable of *very fine* reconstructions where artifacts simply become invisible, see Figure 6.

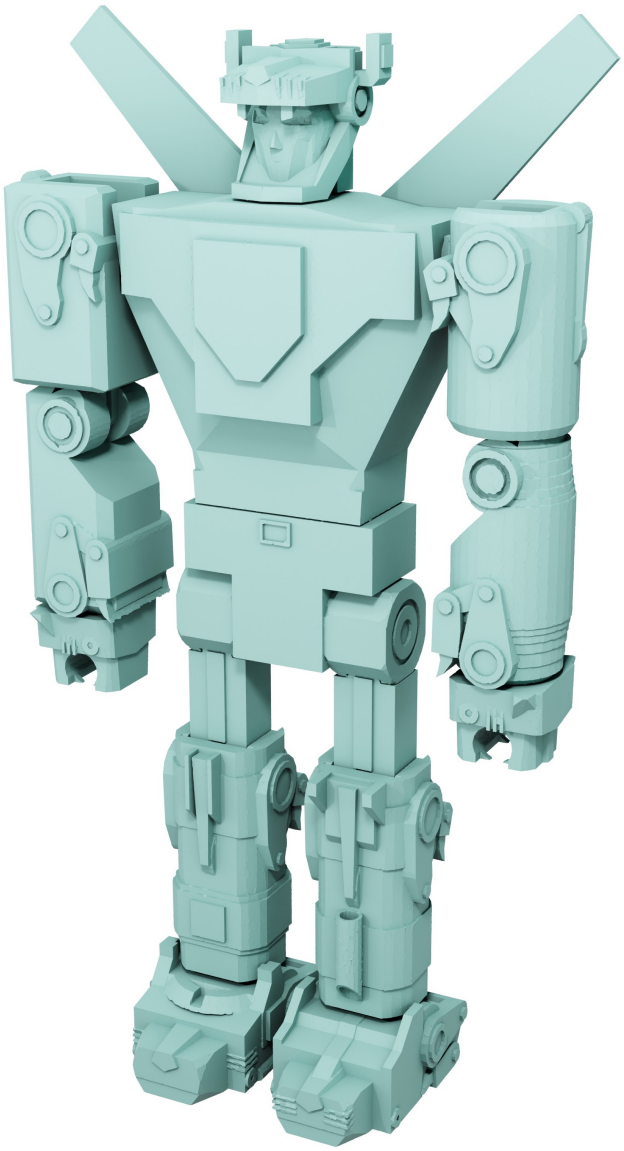


Figure 6: Direct optimization on a 256^3 grid.

5. Comparison to Deep Marching Tetrahedra

For completeness, we provide additional analysis and comparison of our approach to Deep Marching Tetrahedra [7] (DMT). Several critical differences exist between DMT and our Voromesh:

- the output connectivity of DMT meshes is fixed by the template, while ours is flexible and determined by construction of a corresponding Voronoi diagram;
- the representation of DMT is hybrid as a MLP is necessary to model a continuous implicit field of SDF and displacements, whereas our VoroMesh can be used directly without neural networks.
- DMT does not provide strong topological guarantees, namely the absence of self-intersections.

In the absence of an official implementation from the DMT paper, we rely on a tutorial code from a separate NVIDIA library[1], which overfits a single MLP to a given shape — a setting similar to our direct optimisation experiment. It relies on a tetrahedra grid of size 128, which has roughly the same number of vertices as a voxel grid of resolution 65 (we used 64 in our experiment). Using it out-of-the-box yields poor results, see Figure 7a. We believe it can be attributed to the underlying hybrid model; more precisely, to the inability of the MLP to represent high-frequency details as explained in [8]. We also tried to optimize the tetrahedra vertices displacement and signed distance predictions directly; because the parameters are now independent and no longer predicted by an MLP, the initial unit sphere fitting leads to worse results, see Figure 7b: the

(a) DMT (b) DMT-d1 (c) DMT-d2 (d) Ours

Figure 7: Visual comparison between DMT (tetrahedra grid of size 128^3) and our method (grid of size 64^3). Left to right: vanilla DMT version, DMT with direct optimisation, DMT with direct optimisation and ground-truth SDF initialization, and VoroMesh.

Method	CD	F1	NC
DMT (128-d2)	0.712	0.926	0.962
VoroMesh (64)	**0.645**	**0.938**	**0.975**

Table 1: Quantitative comparisons for an optimization-based 3D reconstruction task on the Thingi32 dataset.

[1] DMT tutorial.

loss function is unable to optimize the tetrahedra situated far from the target surface. To alleviate this problem, we further help DMT by initializing the displacements to zero and the predicted signed distance to the ground-truth SDF with respect to the original surface; but resulting reconstructions still lack surface smoothness and finer details and are outperformed by our VoroMesh representation, see Figure 7c and Table 1.

6. Additional timings

We now provide timings for optimization-based and learning-based experiments. Our mesh extraction, which relies on CGAL, is very fast (Table 2), while our inference time for the learning-based experiment is comparable to state-of-the-art methods (Table 3).

Grid Size	Mesh Extraction (s)	Full Execution (s)
32	0.03	5.0
64	0.11	10.2
128	0.46	57.3

Table 2: Mean timings on the Thingi32 dataset for optimization-based 3D reconstruction. Full execution timings include mesh extraction.

Method	Grid Size	Mean Inference (s)
NDC [2]	32	0.05
NMC [3]	32	0.18
Ours	32	0.19
NDC [2]	64	0.12
NMC [3]	64	0.97
Ours	64	0.63

Table 3: Mean timings on the ABC test set for learning-based 3D reconstruction.

Figure 8: VoronoiNet [9] (top row), our method (middle row) and target shape (bottom row) for a grid of size 32^3

(a) MC [5] (b) DC [4] (c) SAP [6] (d) Ours (e) Ground Truth

Figure 9: Visual comparison of optimization-based methods for grids of size 32^3 (top row), 64^3 (middle row), and 128^3 (bottom row)

(a) MC [5] (b) DC [4] (c) SAP [6] (d) Ours (e) Ground Truth

Figure 10: Visual comparison of optimization-based methods for grids of size 32^3 (top row), 64^3 (middle row), and 128^3 (bottom row)

(a) MC [5] (b) DC [4] (c) SAP [6] (d) Ours (e) Ground Truth

Figure 11: Visual comparison of optimization-based methods for grids of size 32^3 (top row), 64^3 (middle row), and 128^3 (bottom row)

(a) MC [5] (b) DC [4] (c) SAP [6] (d) Ours (e) Ground Truth

Figure 12: Visual comparison of optimization-based methods for grids of size 32^3 (top row), 64^3 (middle row), and 128^3 (bottom row)

(a) NMC [3] (b) NDC [2] (c) Ours (d) Ground Truth

Figure 13: Visual comparison of learning-based methods for grids of size 32^3 (top row), 64^3 (bottom row)

(a) NMC [3] (b) NDC [2] (c) Ours (d) Ground Truth

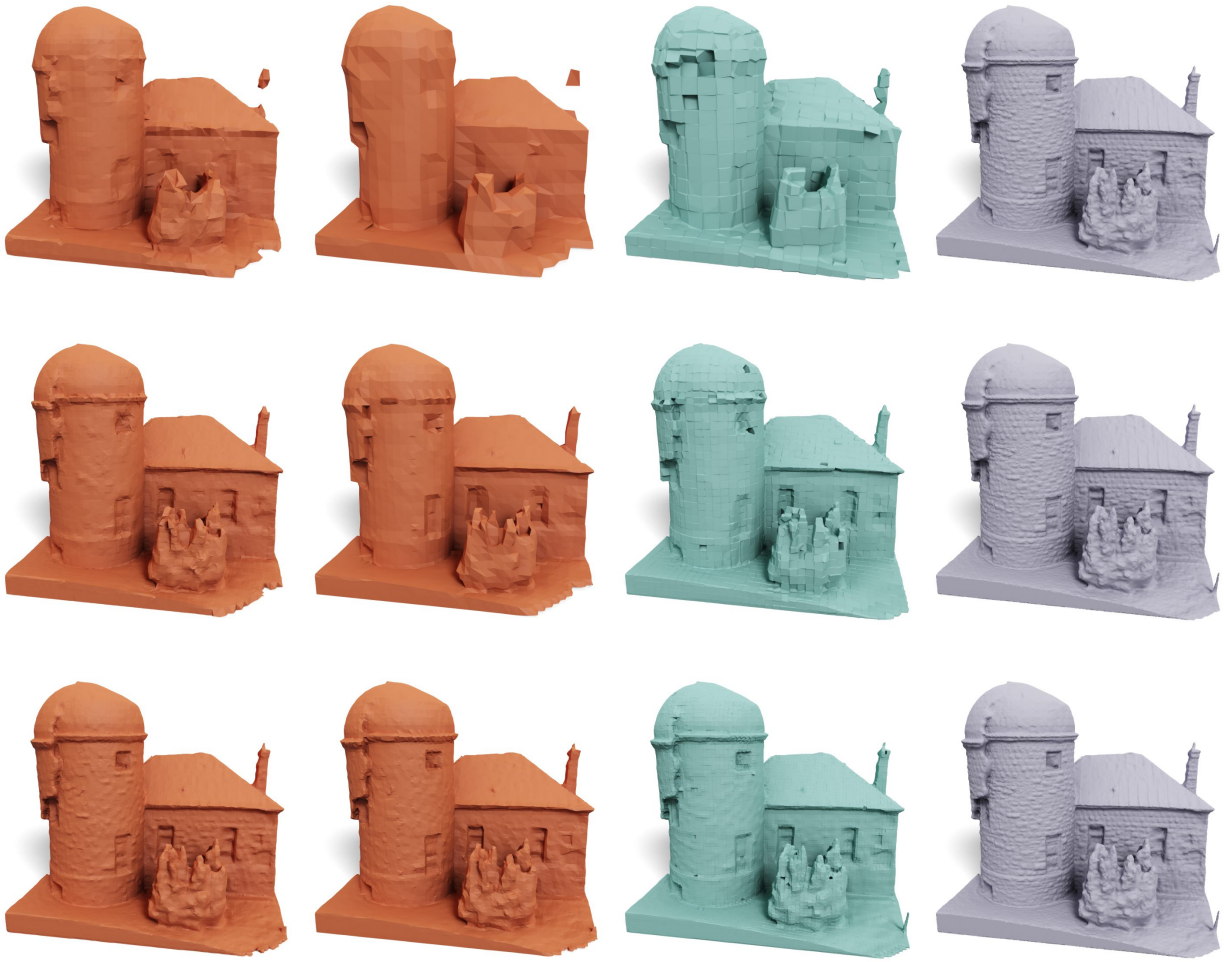
Figure 14: Visual comparison of learning-based methods for grids of size 32^3 (top row), 64^3 (bottom row)

(a) NMC [3] (b) NDC [2] (c) Ours (d) Ground Truth

Figure 15: Visual comparison of learning-based methods for grids of size 32^3 (top row), 64^3 (bottom row)

(a) NMC [3] (b) NDC [2] (c) Ours (d) Ground Truth

Figure 16: Visual comparison of learning-based methods for grids of size 32^3 (top row), 64^3 (middle row), and 128^3 (bottom row)



(a) NMC [3] (b) NDC [2] (c) Ours (d) Ground Truth

Figure 17: Visual comparison of learning-based methods for grids of size 32^3 (top row), 64^3 (middle row), and 128^3 (bottom row)

References

[1] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991. 1

[2] Z. Chen, A. Tagliasacchi, T. Funkhouser, and H. Zhang. Neural dual contouring. *ACM Transactions on Graphics (TOG)*, 41(4), 2022. 2, 4, 10, 11, 12, 13, 14

[3] Z. Chen and H. Zhang. Neural marching cubes. *ACM Transactions on Graphics (TOG)*, 40(6), 2021. 2, 4, 10, 11, 12, 13, 14

[4] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of Hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002. 6, 7, 8, 9

[5] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 6, 7, 8, 9

[6] S. Peng, C. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger. Shape as points: A differentiable Poisson solver. In *NeurIPS*, 2021. 6, 7, 8, 9

[7] T. Shen, J. Gao, K. Yin, M.-Y. Liu, and S. Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *NeurIPS*, 2021. 1, 3

[8] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020. 3

[9] F. Williams, J. Parent-Levesque, D. Nowrouzezahrai, D. Panozzo, K. M. Yi, and A. Tagliasacchi. VoronoiNet: General functional approximators with local support. In *CVPR Workshop*, 2020. 5