Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis

Tianchang Shen [1,2,3] **Jun Gao**[1,2,3] **Kangxue Yin** [1]

Ming-Yu Liu [1] **Sanja Fidler**[1,2,3]

NVIDIA[1] University of Toronto[2] Vector Institute[3]

{frshen, jung, kangxuey, mingyul, sfidler}@nvidia.com

In the supplementary material, we first extensively analyze the differentiability of Marching Tetrahedra and compare with MeshSDF [16] in Section A. We provide further details about the network architecture of DMTET in Section B, followed by details about our training algorithm in Section C. Experimentally, we provide additional comparisons and ablation study in Section D, as well as quantitative and qualitative results in Section E. Details about the user study are included in Section F. Limitations, failure cases, and future work are discussed in Sec. G. We additionally provide 360-degree renderings of our results (see *demo.mp4*), the visualizations of all animal models in TurboSquid dataset (see Folder *TurboSquid*), and the visualization of results on all test shapes of TurboSquid (see *voxel_upsample.html* in the Supplementary Material.

A Differentiability of Marching Tetrahedra

The Marching Tetrahedra (MT) [4] layer plays a core role in our representation, converting the Signed Distance Field encoded by the tetrahedral grid to an explicit triangular mesh. In the main paper, we have demonstrated how MT determines the surface typology inside the tetrahedron (the number of triangular faces and how vertices are connected), and compute the vertex location of the extracted iso-surface as in Fig 3. The gradient from a loss defined on the extracted iso-surface L_{surf} can be back-propagated to both, vertex positions and the SDF values via the chain rule:

$$\frac{\partial L_{surf}}{\partial v_a} = \frac{\partial L_{surf}}{\partial v'_{ab}} \frac{s(v_b)}{s(v_b) - s(v_a)}, \quad \frac{\partial L_{surf}}{\partial s(v_a)} = \frac{\partial L_{surf}}{\partial v'_{ab}} \frac{s(v_b)(v_a - v_b))}{(s(v_b) - s(v_a))^2}, \tag{1}$$

which are continuous when $\text{sign}(s(v_b)) \neq \text{sign}(s(v_a))$. Note that our observation not only applies to MT, but also applies to MC, since both algorithms use the same way of extracting the zero crossing point.

To support our argument, we follow the experimental settings in MeshSDF [16] and extensively evaluate our MT layer. We first show the capability to support the differentiable topology changes via the MT layer, and then compare with MeshSDF [16] on the task of single-image 3D reconstruction.

Differentiable Topology Changes Following MeshSDF [16], we optimize a deep implicit network [14] to transform the SDF of a sphere to the SDF of a desired object with a different typology, in particular, torus and bunny rabbit. The objective function is the Chamfer Distance between the ground truth mesh and the extracted surface from SDF using the MT layer. Qualitative results in Fig. A demonstrate that MT can effectively propagate the loss defined on the surface to the implicit field and change the typology of the surface through our MT layer.

Quantitative Comparison with MeshSDF [16] on Single-Image 3D Reconstruction We follow the experimental setting from MeshSDF [16] on the task of Single-Image 3D Reconstruction. Specifically, we first train a 3D reconstruction model, which takes a latent code z predicted from the

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

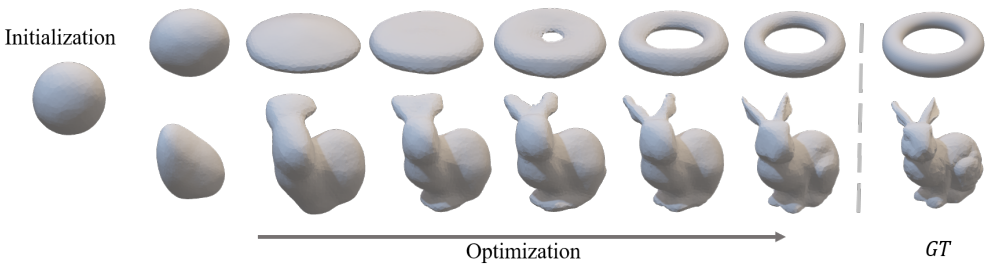

Figure A: Optimizing an implicit field parametrized by MLPs to minimize Chamfer Distance between the GT mesh and the extracted mesh from the SDF through our MT layer.

input image via an image encoder and outputs a signed distance value for continuous 3D locations, supervised only by the SDF loss. During the inference time, the encoder predicts an initial latent code z, and we further optimize z to minimize the silhouette loss via differentiable rasterization. We refer the readers to original paper [16] for more details. Both MeshSDF and MT allow gradients defined on the extracted mesh to propagate back to z to reconstruct better shape. For this experiment, we use the official code[1] and replace their differentiable iso-surfacing layer with MT. With optimization at inference time, MeshSDF reports an improvement of 5.82% on Chamfer Distance evaluated on ShapeNet cars. MT achieves an improvement of 7.75% on the same subset. We further jointly optimize z and positions of grid vertices, since MT is defined on a deformable grid – we achieve an improvement of 8.33%, significantly outperforming MeshSDF [16]. Note that, we only use MT to optimize the deep implicit field, and conduct evaluation on surface extracted with MC at high resolution (256), same as MeshSDF.

B Network Architectures

Input Encoder Given an input point cloud x, we use PVCNN [11] to extract multi-scale 3D feature volumes of sizes: $R_1^3 \times C_1$,$R_2^3 \times C_2$ and $R_3^3 \times C_3$, where the spatial resolution $R_1 = 32, R_2 = 16, R_3 = 8$ and the number of channels $C_1 = 64, C_2 = 256, C_3 = 512$. To extract the point-wise feature for a point location $v \in \mathbb{R}^3$, we use trilinear interpolation to obtain the feature vector from each 3D feature volume, and concatenate these vectors together to form the final feature vector $F_{vol}(v, x) \in \mathbb{R}^{832}$. Similarly to DefTet [5], we use two encoders: one provides features for initial prediction of SDF and the other provides features for surface refinement and surface subdivision.

MLPs for Initial Prediction of SDF For the initial prediction of SDF, we employ a four-layer MLPs with hidden dimensions 256, 256, 128 and 64, respectively. We extract the activations before the output layer, denoted as $f(v)$, and pass them to the surface refinement step.

Surface Refinement For the surface refinement network, we use a 2-layer Graph-Convolutional Network [10] (GCN) followed by a 2-layer MLPs. Specifically, given a graph $G = (V_{surf}, E_{surf})$ where V_{surf} and E_{surf} are vertices and edges in surface tetrahedrons T_{surf} with per-vertex feature $f_{v_i}^0$, we use Graph-ResNet layers to predicted the residual to $f_{v_i}^l$ as:

$$r_{v_i}^l = ReLU(w_0^l f_{v_i}^l + \sum_{v_j \in \mathcal{N}(v_i)} w_1^l f_{v_j}^l) \quad (2)$$

$$r_{v_i}^{l+1} = w_2^l r_{v_i}^l + \sum_{v_j \in \mathcal{N}(v_i)} w_3^l r_{v_j}^l \quad (3)$$

$$f_{v_i}^{l+1} = ReLU(f_{v_i}^l + r_{v_i}^{l+1}) \quad (4)$$

where $\mathcal{N}(v_i)$ denotes vertices connect to v_i in G and w_0^l to w_3^l are weight matrices for the residual layer l. Next, we use fully connected layers taking $f_{v_i}^{l+1}$ as input to predict the per-vertex position offset Δv_i, SDF residual value $\Delta s(v_i)$ and updated feature vector $f(v_i)$. The dimensions of the

[1] https://github.com/cvlab-epfl/MeshSDF

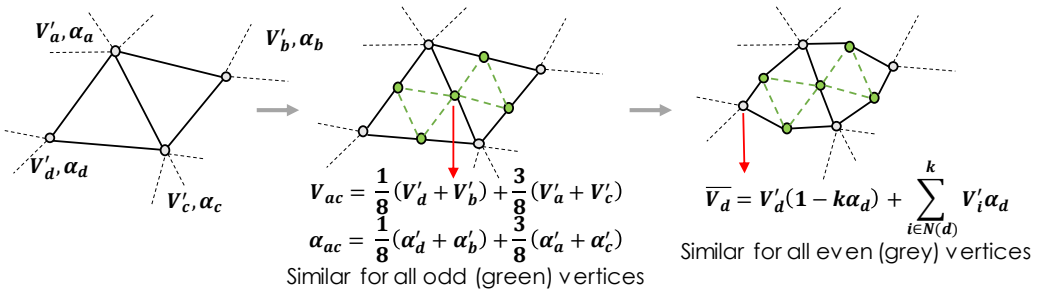

Figure B: Scheme of Learnable Surface Subdivision. We build our module on top of Loop Subdivision [12], and additionally predict the α to control the smoothness of the parametric surface.

GCN layers are 256 and 128, respectively. The dimensions of the MLPs are 128 and 64, respectively. Note that during the volume subdivision step, the per-vertex feature of the midpoint is obtained by averaging the feature vectors along the edge:

$$f(v'_{ac}) = \frac{1}{2}(f(v_a) + f(v_c)) \tag{5}$$

Learnable Surface Subdivision Given the predicted surface extracted using MT, we further employ a separate GCN which shares the same network architecture as the one in the surface refinement step, to predict the updated position of each mesh vertex v'_i and α_i. We then follow the scheme of the Loop Subdivision method [12] as illustrated in Fig. B. Note that α_i is also interpolated for new (odd) vertices and carried over to the next subdivision step. This subdivision step is applied iteratively to approximate a parametric surface.

3D Discriminator We provide details about how we compute the discretized SDF, $S_{real} \in \mathbb{R}^{N \times N \times N}$ and $S_{pred} \in \mathbb{R}^{N \times N \times N}$, as input to D. For the predicted mesh M_{pred} and the corresponding ground truth shape M_{gt}, we first compute the vertex curvature for all vertices in M_{gt}. Next, we sample a vertex v from vertices with magnitude of curvature larger than a threshold. This sampling step is necessary because most of the surface of the animal is smooth. Training D on all surface samples leads to smooth results without high-frequency details. We add a small noise to v to avoid overfitting, and compute the ground truth signed distance field S_{real} around v. Specifically, let $n = (i, j, k) \in \mathbb{N}^3$ be the multi-index with i, j, k corresponds to the 3 dimensions of S_{real}. We compute the SDF in each grid position as:

$$S_{real_n} = SDF(v + \frac{(n - N/2)}{r}, M_{gt}) \tag{6}$$

where r is a hyperparameter controling the range of the discretized SDF. If r is larger, S_{real} occupies a smaller region around v, and vise versa. The S_{pred} is computed in the same location with respect to the predicted mesh:

$$S_{pred_n} = SDF(v + \frac{(n - N/2)}{r}, M_{pred}). \tag{7}$$

We follow DecorGAN [2] and use the discriminator architecture shown in Fig. C. Besides the discretized SDF, the discriminator additionally takes the feature vector $F_{vol}(v, x)$ as input, which is extracted from the highest-level feature volume at position v. This feature vector provides global semantic information to the discriminator.

C Experimental Details

In this section, we first describe the dataset preparation procedure and generator details as they are the same for the two applications in the main paper. Next, we discuss specific experiment details for each application.

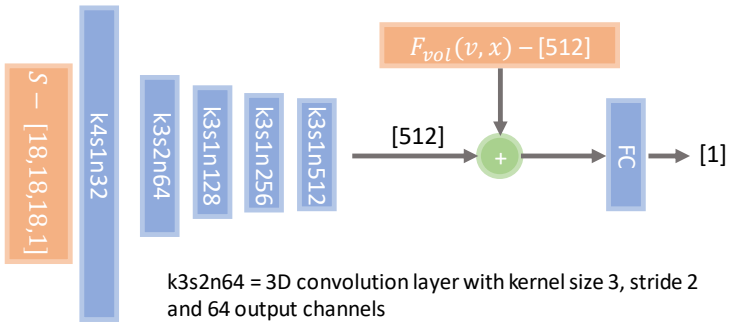

Figure C: The architecture of discriminator. Numbers in brackets correspond to the size of the tensor.

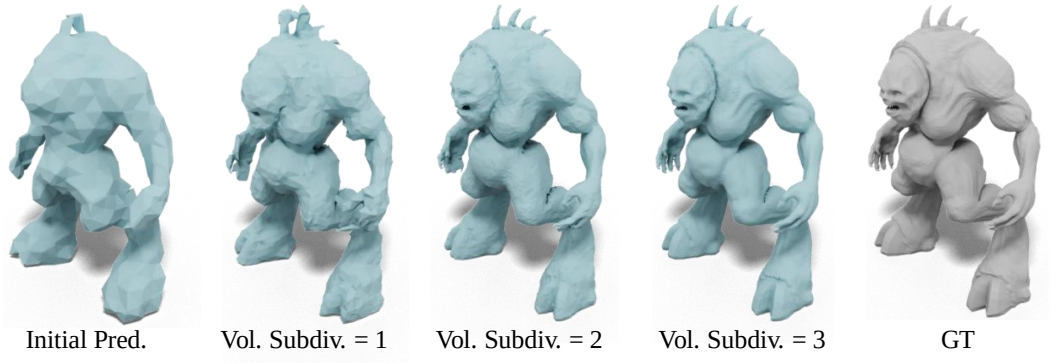

Figure D: An example of overfitting a complex shape with DMTET by applying multiple subdivision steps. The loss is only computed on the final surface (Vol.Subdiv.=3) and propagated back to optimize the typology at all subdivision levels. The initial resolution is 40 (which has the same number of vertices as a voxel grid of resolution of 20) and each volume subdivision step doubles the spatial resolution.

C.1 Dataset Preparation

We provide visualizations of all animal models in TurboSquid dataset in the Supplementary Material (see Folder *TurboSquid*). We manually checked all the animal models we used and did not find personally identifiable information or offensive content in our data.

Since many models in TurboSquid and ShapeNet are not watertight, we follow the pipeline in Kaolin [8] to convert all objects to watertight meshes. More specifically, each object is first voxelized at high resolution. We use a resolution of 256 and 100 for TurboSquid and ShapeNet objects, respectively. We use lower resolution for ShapeNet as it contains a lot of thin structures (e.g. wire, airfoil), which degenerates the performance of occupancy-based baselines. We voxelize these shapes at lower resolution to thicken the thin structures as in OccNet [13]. The resulting voxel is then converted to a surface mesh with Marching Cubes followed by Laplacian Smoothing [7].

C.2 Generator

We use the same generator for 3D shape synthesis from voxels and point cloud reconstruction. Unless otherwise noted, we start with an initial tetrahedral grid of resolution 70, which has 47,416 vertices (same as the number of vertices in a voxel grid of resolution 35) and 251,447 tetrahedrons. After the initial prediction of SDF on grid vertices, we perform a surface refinement step. Then, we apply volume subdivision once followed by another surface refinement step. Note that the weights are not shared for the two surface refinement modules. We found conducting more than a single volume subdivision step does not further improve the accuracy when reconstructing objects in our datasets. Nevertheless, volume subdivision can be applied iteratively to reconstruct high-resolution objects, as shown in Fig D.

For surface subdivision, we only apply it once during training and compute loss on the resulting surface. At inference time, we subdivide twice for visualization and evaluation. The surface subdivision step can be applied iteratively to obtain the parametric surface at "infinite resolution", but we found subdividing twice already approximates the final surface well.

C.3 3D Shape Synthesis from Coarse Voxels

Input to a Discriminator To sample high-curvature vertices from the ground truth mesh, we compute the vertex defects, which equals 2π minus the sum of the angles of all faces connecting each vertex. We set the threshold to be $\frac{\pi}{16}$ which we found covers most regions containing details we want to synthesize. We set $N = 18$ and $r = 128$ following DecorGAN [2]. We sample 10 patches for each input and GT pair.

Training Details For the SDF loss, we truncate the ground truth SDF value at $\pm$ 0.03 to focus on surface, and evaluate it only at vertices of non-surface tetrahedrons before volume subdivision as well as surface tetrahedrons after volume subdivision. This ensures that the ground truth SDF value of deformed vertices are computed on their updated locations. We set hyper-parameters $\lambda_{\text{cd}} = 500$, $\lambda_{\text{normal}} = 1e-6$, $\lambda_{\text{def}} = 1$, $\lambda_{\text{SDF}} = 0.4$ and $\lambda_{\text{G}} = 10$. We use the Adam Optimizer with the learning rate equal to $1e-4$ for both the generator and the discriminator. Although all modules in DMTET can be trained end-to-end, we found that computing the loss on the mesh obtained after surface subdivision is slow in practice due to the large number of faces. Thus, we first train DMTET without surface subdivision, then fine-tune the surface subdivision module added to the pretrained model. In particular, we initialize the generator by minimizing the SDF loss for 500 iterations and then train it on all reconstruction losses for 10k iterations. Next, we add the discriminator D and perform adversarial learning for another 20k iterations. Finally, we add the surface subdivision module and train our complete model for 30k iterations. The total training takes approximately 6 days on 4 Nvidia V100 GPUs.

Evaluation Metrics For this task, we evaluate our models and baselines using Chamfer Distance, Chamfer-L1, normal consistency score, light field distance (LFD) [1] and classification score (Cls). We follow DefTet [5] to calculate the Chamfer and Chamfer-L1:

$$\text{Chamfer}(\mathcal{F}_{\text{Pred}}, \mathcal{F}_{\text{GT}}) = \frac{1}{2|\partial\mathcal{F}_{\text{Pred}}|} \sum_{p \in \partial\mathcal{F}_{\text{Pred}}} \min_{q \in \partial\mathcal{F}_{\text{GT}}} ||p-q||_2 + \frac{1}{2|\partial\mathcal{F}_{\text{GT}}|} \sum_{p \in \partial\mathcal{F}_{\text{GT}}} \min_{q \in \partial\mathcal{F}_{\text{Pred}}} ||p-q||_2, \quad (8)$$

$$\text{Chamfer-L1}(\mathcal{F}_{\text{Pred}}, \mathcal{F}_{\text{GT}}) = \frac{1}{|\partial\mathcal{F}_{\text{Pred}}|} \sum_{p \in \partial\mathcal{F}_{\text{Pred}}} \min_{q \in \partial\mathcal{F}_{\text{GT}}} ||p-q||_1 + \frac{1}{|\partial\mathcal{F}_{\text{GT}}|} \sum_{p \in \partial\mathcal{F}_{\text{GT}}} \min_{q \in \partial\mathcal{F}_{\text{Pred}}} ||p-q||_1, \quad (9)$$

where $\mathcal{F}_{\text{GT}}, \mathcal{F}_{\text{Pred}}$ denote ground truth and prediction, respectively, and $\partial\mathcal{F}_{\text{GT}}, \partial\mathcal{F}_{\text{Pred}}$ denote the set of sampled points on the surface of ground truth and the prediction, respectively. The normal consistency score is computed via Eqn.3 in the main paper. LFD [1] measures the visual similarity of two shapes with image differences in light fields. We use the official implementation[2] of LFD [1]. For Cls score, we follow the evaluation pipeline from DecorGAN [2]. More specifically, we convert all predicted shapes and GT models to high resolution (256^3) voxels and render 24 images from random views for each shape. Next, we train a ResNet using images of GT shapes as real examples and predicted shapes as fake examples. The rendered images have size 256 x 256 and are randomly cropped to 64 x 64 patches before feeding into the image classifier. We report the mean classification accuracy as Cls score.

C.4 Point Cloud Reconstruction

Training Details We use hyper-parameters $\lambda_{\text{cd}} = 500$, $\lambda_{\text{normal}} = 1e-6$, $\lambda_{\text{def}=}1$ and $\lambda_{\text{SDF}} = 0.4$. Similarly, we truncate the ground truth SDF value, and first initialize the network by training only with the SDF loss for 500 iterations to avoid computing the surface loss on random surfaces. We then train the model for 20k iterations without surface subdivision, and another 20k iterations with surface subdivision. The training takes approximately 4 days on 4 Nvidia V100 GPUs.

[2]https://github.com/kacperkan/light-field-distance

	L2 Chamfer ↓	L1 Chamfer ↓	Norm. Consistency ↑
ConvOnet [15]	0.83	2.41	0.901
ConvOnet [15] - Our Arch.	0.79	2.30	0.912
DMTET wo (Adv,Surf.,Surf Loss)	0.79	2.31	**0.915**
DMTET wo (Adv.,Surf.)	**0.76**	**2.20**	**0.914**
DMTET	**0.75**	**2.19**	**0.918**

Table A: Additional Comparisons on 3D Shape Synthesis from Coarse Voxels. Comparing with ConvOnet [15] using our architecture, our DMTET achieves significant improvements. Comparing DMTET wo (Adv., Surf., Surf loss) with DMTET wo (Adv., Surf.), adding surface loss significantly improves the performance.

Evaluation Metrics We evaluate different methods using several metrics: 3D Intersection-Over-Union (IOU), Chamfer Distance, Chamfer-L1 and F-score. For 3D IOU, we follow the pipeline introduced in OccNet [13] and randomly sample 100k points in the 3D space to evaluate ground truth occupancies and check whether the points are inside or outside of the predicted mesh. The definitions of Chamfer and Chamfer-L1 Distance follow Deftet [5]. The F1 score is calculated as

$$F_1 = 2 \cdot \frac{\text{completeness} \cdot \text{accuracy}}{\text{completeness} + \text{accuracy}} \tag{10}$$

where completeness is the percentage of points in $\partial\mathcal{F}_{\text{GT}}$ for which at least one point from $\partial\mathcal{F}_{\text{pred}}$ is within a radius of 0.01. Similarly, accuracy is the percentage of points in $\partial\mathcal{F}_{\text{pred}}$ for which at least one point from $\partial\mathcal{F}_{\text{GT}}$ is within the threshold.

For clarity, we multiply 3D IoU, Chamfer Distance, Chamfer-L1 and F-score by 100 when reporting them in the tables.

D Additional Comparisons

We provide additional comparisons on 3D Shape Synthesis from Coarse Voxels. We first compare with ConvOnet [15] using our network architecture, and further compare with our own version of SDF prediction without training on the surface.

Comparison with ConvOnet [15] using our network architecture We use points sampled from the coarse voxel after MC as inputs to ConvOnet [15] and DMTET. Since ConvOnet [15] uses a different 3D network architecture to predict occupancies from the input points, we additionally provide results of predicting continuous occupancy function with our network architecture. More specifically, we use the PVCNN as a 3D encoder and our MLPs to decode occupancy values, and follow the same training pipeline as in ConvOnet [15]. We denote this model as ConvOnet [15]-Our Arch. Quantitative and qualitative results are shown in Table A and Fig. F, respectively. Our DMTET still reconstructs higher quality shapes with more details, and outperforms ConvOnet [15]-Our Arch. in terms of L1/L2 Chamfer distances and Normal Consistentcy loss, demonstrating the stronger capability of our model.

Ablating the importance of a surface loss To further illustrate the effect of learning using the loss defined on the resulting surface, we ablate DMTET by removing the losses defined on the surface and train the model purely using SDF loss (Eqn. 5 in the main paper). In particular, we predict SDF for each vertex in the hierarchical tetrahedral grid, and supervise with SDF loss. During inference, we extract the mesh using MT. We denote this model as DMTET wo (Adv. Surf. Surf-loss.). For a fair comparison, we additionally train our model with surface loss and remove the adversarial loss and surface subdivision. The quantitative and qualitative comparisons are shown in Table A and Fig. F, respectively. Comparing these two methods, applying surface loss improves the reconstruction accuracy and generates shapes with more geometric details and fewer artifacts, demonstrating the effectiveness of directly optimizing for the reconstructed surface.

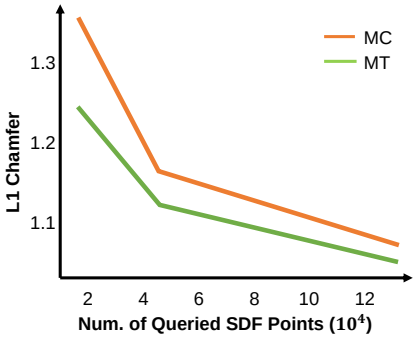

Figure E: Comparing MT vs. MC in Point Cloud 3D Reconstruction, evaluated on ShapeNet Chairs.

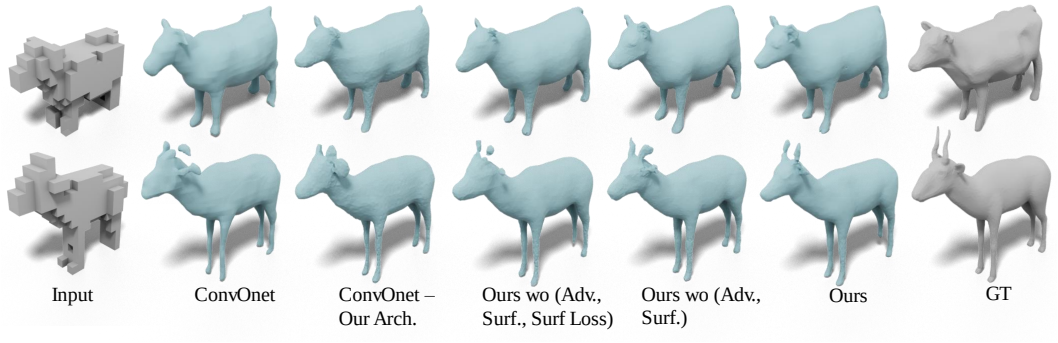

Figure F: Additional comparisons on 3D shape Synthesis from Coarse Voxels. Comparing with ConvOnet [15] using our architecture, our DMTET reconstructs shapes with more geometric details. Comparing DMTET wo (Adv., Surf., Surf loss) with DMTET wo (Adv., Surf.), adding surface loss helps predicting more visually pleasing shape with more details and fewer artifacts.

Category	Airplane	Bench	Dresser	Car	Chair	Display	Lamp	Speaker	Rifle	Sofa	Table	Phone	Vessel	**Mean**↓
ConvOnet[15]	1.63	2.06	2.19	1.88	2.10	2.06	2.59	2.31	4.74	2.11	2.12	1.73	2.17	2.28
DMTET	1.30	1.70	2.01	1.72	1.93	1.96	1.89	2.18	4.43	1.96	1.97	1.68	1.88	2.04

Table B: Quantitative Results on **3D Shape Synthesis from Coarse Voxels on ShapeNet** (Chamfer L1).

Comparison of MT against MC in Point Cloud 3D Reconstruction While in Sec. 4.2.1 in the main paper, we compare the oracle performance of MC with MT at different resolution, here we additionally provide comparison in a learning setup, where we train a neural network to predict the SDF values at the grid vertices and running MC/MT to obtain the final surface using point cloud as input on ShapeNet chairs. The loss function and network architectures are the same for both MC and MT, and we do not deform the vertices or use GCN for surface refinement here for a fair comparision with MC. As summarized in Fig. E, MT consistently outperforms MC when querying the same number of points, agreeing with our analysis in 4.2.1.

The runtime of MT/MC is comparable at same resolution (measured by number of vertices in grid). At 4.5k vertices, it takes 16.5 ms to inference SDF values with network, and 7 ms to run MT/MC for meshing. Considering the higher performance of MT at the same resolution, we argue MT is a better choice for learning.

E Additional Experimental Results

E.1 3D Shape Synthesis from Coarse Voxels

We provide more qualitative examples in Fig. G. Compared to all baselines, our DMTET generates shapes with finer geometric details and fewer artifacts. In addition, we provide visualization of results on all test shapes in the Supplementary Material (see *voxel_upsample.html*).

In addition, we evaluate our method on ShapeNet dataset and summarize the result in Tab. B and Fig. I. Note that we do not apply adversarial loss in this case because our discriminator is designed to capture surface details while in ShapeNet most details are structural (e.g. holes in wheels.). Without adversarial learning, our model still outperforms ConvONet [15] across all object categories with finer details synthesised.

E.2 Point Cloud 3D Reconstruction

Table C, D, E show quantitative results in terms of 3D IOU, Chamfer Distance, and F-score, respectively. We also provide more qualitative examples in Fig. H. We achieve significant improvements over all baselines in terms of all metrics. Qualitatively, our DMTET generates shapes with more details and correct topology.

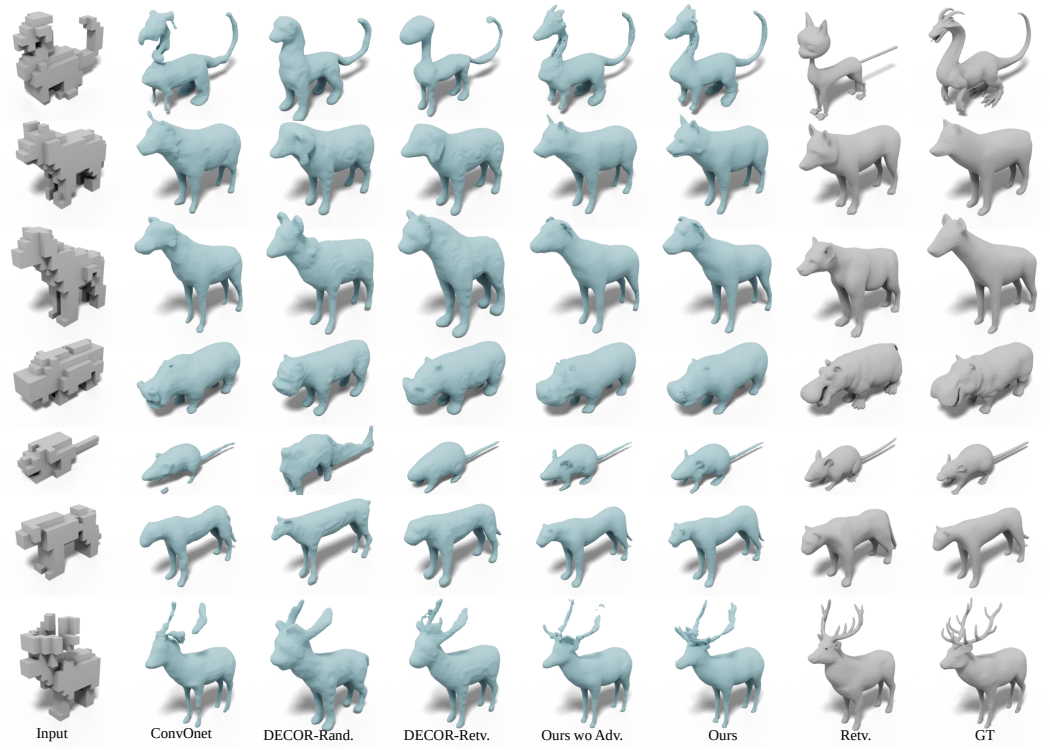

Figure G: **Qualitative results on 3D shapes Synthesis from Coarse Voxels.** Compared to all baselines, our method reconstructs shapes with much higher quality. Adding GAN further improves the realism of the generated shape. We also show the retrieved shapes from the training set in the second last column

Category	Airplane	Bench	Dresser	Car	Chair	Display	Lamp	Speaker	Rifle	Sofa	Table	Phone	Vessel	**Mean**↑	**Time**(ms)↓
3D-R2N2 [3]	70.75	62.98	85.44	86.11	74.83	77.53	63.75	88.11	63.38	85.16	70.37	80.66	76.56	75.82	174
DMC [9]	70.73	64.79	87.48	85.03	79.04	83.34	61.77	89.63	64.33	88.88	78.29	90.09	75.57	78.38	349
Pixel2mesh [18]	84.18	72.06	86.39	91.01	76.02	85.25	65.43	89.90	78.37	90.38	73.67	92.94	83.58	82.24	**30**
ConvOnet [15]	86.91	80.91	93.41	92.78	88.23	91.11	79.72	94.21	81.05	93.99	89.21	94.77	87.57	88.76	866
MeshRCNN [6]	84.33	76.52	90.96	91.40	85.52	89.23	74.79	92.77	78.18	92.56	86.04	93.86	84.96	86.24	228
DEFTET [5]	85.51	79.88	93.23	92.05	87.12	90.74	77.98	93.80	76.68	93.66	88.46	94.50	86.34	87.69	61
DMTET wo (Def, Vol., Surf.)	85.93	78.25	92.94	93.20	87.95	90.50	78.49	94.76	77.94	93.87	87.60	94.44	87.51	87.95	52
DMTET wo (Vol., Surf.)	89.09	82.83	94.04	93.69	89.45	92.84	82.62	95.11	83.78	94.61	90.49	95.69	89.28	90.27	52
DMTET wo Vol.	89.83	83.93	94.56	93.88	90.06	92.91	83.04	95.35	84.92	95.11	91.36	93.62	89.74	90.64	67
DMTET wo Surf.	90.24	84.32	**94.74**	**93.89**	90.24	93.02	**83.11**	95.29	**85.20**	95.20	91.31	**97.07**	89.96	**91.05**	108
DMTET	**90.36**	**84.63**	94.55	93.88	**90.33**	**93.28**	**83.11**	**95.31**	84.98	**95.29**	**91.74**	96.21	**89.99**	**91.05**	129

Table C: Quantitative Results on **Point Cloud Reconstruction** (3D IoU).

Category	Airplane	Bench	Dresser	Car	Chair	Display	Lamp	Speaker	Rifle	Sofa	Table	Phone	Vessel	**Mean**↓	**Time**(ms)↓
3D-R2N2 [3]	0.55	0.59	0.60	0.59	0.61	0.62	0.63	0.63	0.53	0.59	0.66	0.60	0.57	0.60	174
DMC [9]	0.56	0.53	0.45	0.59	0.50	0.44	0.75	0.52	0.54	0.41	0.47	0.30	0.62	0.51	349
Pixel2mesh [18]	0.34	0.47	0.47	0.40	0.76	0.46	0.82	0.54	0.30	0.38	0.73	0.26	0.43	0.49	**30**
ConvOnet [15]	0.27	0.31	0.32	0.37	0.34	0.31	0.44	0.38	0.28	0.30	0.31	0.22	0.33	0.32	866
MeshRCNN [6]	0.30	0.35	0.36	0.39	0.37	0.34	0.41	0.42	0.30	0.33	0.34	0.24	0.36	0.35	228
DEFTET [5]	0.30	0.32	0.33	0.38	0.36	0.32	0.42	0.40	0.33	0.31	0.32	0.23	0.35	0.33	61
DMTET wo (Def, Vol., Surf.)	0.28	0.32	0.31	0.32	0.33	0.29	0.34	0.34	0.28	0.28	0.31	0.22	0.30	0.30	52
DMTET wo (Vol., Surf.)	0.23	0.28	0.30	0.31	0.31	0.27	0.30	0.33	0.22	0.28	0.28	0.21	0.28	0.28	52
DMTET wo Vol.	0.22	0.26	0.29	0.30	0.30	0.27	0.29	0.33	0.21	0.27	0.27	0.21	0.27	0.27	67
DMTET wo Surf.	**0.21**	**0.26**	0.29	0.30	0.30	0.27	**0.28**	**0.32**	**0.21**	**0.26**	**0.27**	0.20	**0.26**	**0.26**	108
DMTET	**0.21**	**0.26**	**0.28**	**0.29**	**0.29**	**0.26**	**0.28**	**0.32**	**0.21**	**0.26**	**0.27**	**0.19**	**0.26**	**0.26**	129

Table D: Quantitative Results on **Point Cloud Reconstruction** (Chamfer L2).

Category	Airplane	Bench	Dresser	Car	Chair	Display	Lamp	Speaker	Rifle	Sofa	Table	Phone	Vessel	**Mean**↑	**Time**(ms)↓
3D-R2N2 [3]	85.36	83.85	86.82	85.49	83.84	83.34	82.43	87.18	86.21	85.63	79.52	81.52	85.45	84.36	174
DMC [9]	84.66	88.04	93.29	86.31	91.31	93.84	79.67	90.87	85.92	94.32	92.57	98.44	83.48	89.44	349
Pixel2mesh [18]	95.22	91.46	92.61	93.92	86.02	93.26	82.77	90.47	96.35	95.68	87.23	98.99	92.52	92.04	**30**
ConvOnet [15]	98.07	97.40	98.05	95.37	97.73	98.26	94.32	96.48	97.51	98.36	98.33	99.64	96.09	97.355	866
MeshRCNN [6]	97.09	96.01	96.94	94.12	96.64	97.37	93.26	94.80	96.68	97.46	97.25	99.48	94.84	96.30	228
DEFTET [5]	97.53	97.24	97.56	95.00	97.19	97.97	94.01	95.37	95.35	98.04	98.04	99.55	95.75	96.81	61
DMTET wo (Def, Vol., Surf.)	98.29	97.86	98.54	97.28	98.40	98.93	96.29	97.82	97.64	98.86	98.71	99.64	97.24	98.11	52
DMTET wo (Vol., Surf.)	99.11	98.59	98.92	97.68	98.71	99.08	97.46	98.09	98.86	99.07	98.99	99.76	97.91	98.63	52
DMTET wo Vol.	99.17	98.90	98.95	97.76	98.78	99.02	97.44	98.09	98.90	99.14	99.02	99.21	98.00	98.64	67
DMTET wo Surf.	**99.25**	98.77	**99.01**	97.84	**98.81**	99.10	**97.66**	**98.11**	**99.04**	99.16	99.11	99.79	98.08	98.75	108
DMTET	**99.25**	**98.79**	98.99	**97.87**	**98.81**	**99.15**	97.63	98.07	99.02	**99.20**	**99.13**	**99.80**	**98.14**	**98.76**	129

Table E: Quantitative Results on **Point Cloud Reconstruction** (F1).

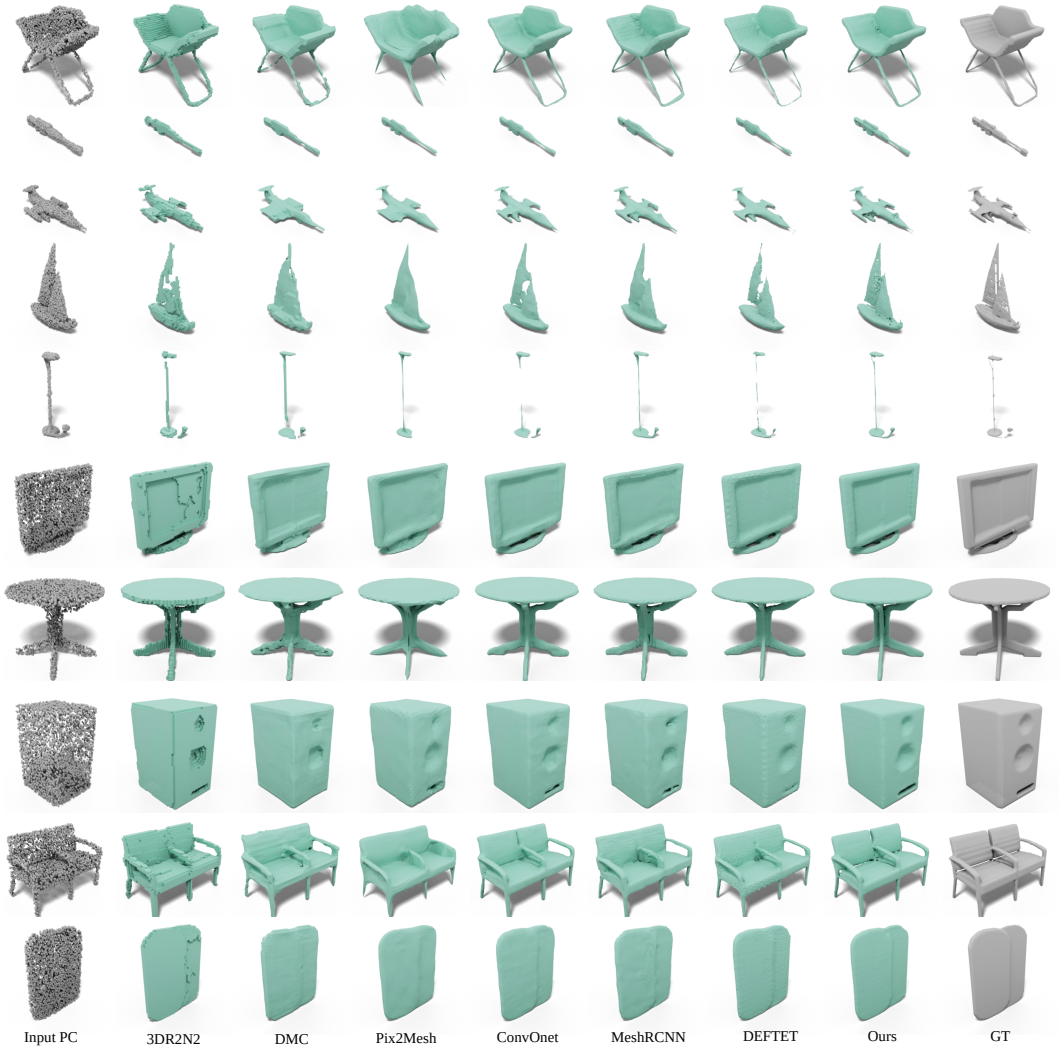

Figure H: **Qualitative results on 3D Reconstruction from Point Clouds:** Our model reconstructs shapes with more geometric details compared to baselines.

E.3 DMTET for Scene Representation

In Fig. J we show an overfitting result to demonstrate that DMTET can scale up to room-level scene.

F User Study Details

The user study reported in Section 4.1 of the main paper was conducted on Amazon Mechanical Turk. The survey form we presented to the users is shown in Figure K. Given two methods that we aim to compare in the study (e.g., DECOR-Retv. vs. DMTet), we generate results of the two methods and render them into videos where the two 3D objects are displayed in a rotating view. The users are asked to answer two questions regarding how the overall shape looks and how realistic are its geometric details, as shown in Figure K. Since we have 442 test shapes in the animal dataset, we generate 442 HITs for comparing the two methods. In each HIT, the order of the two methods displayed to the users are randomly determined to avoid order bias. To reduce outliers in user responses, we ask 3 different users per HIT, and determine the final response via majority voting. Furthermore, a qualified

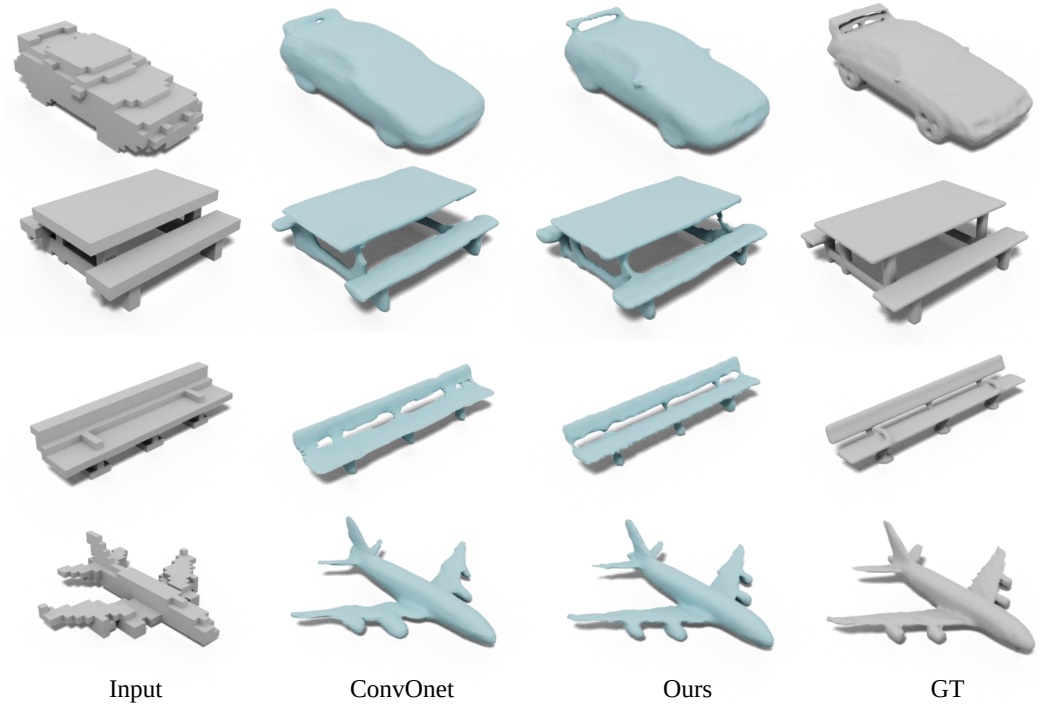

Figure I: **Qualitative results on 3D Shape Synthesis from Coarse Voxels on ShapeNet:** Our model reconstructs shapes with more geometric details and less artifacts compared to ConvONet [15]

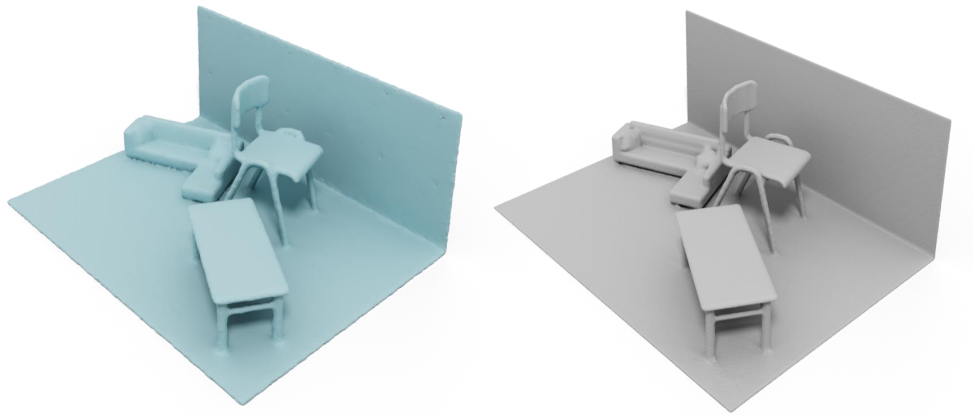

Figure J: We demonstrate DMTET can scale up to room-level scene in an overfitting setting.

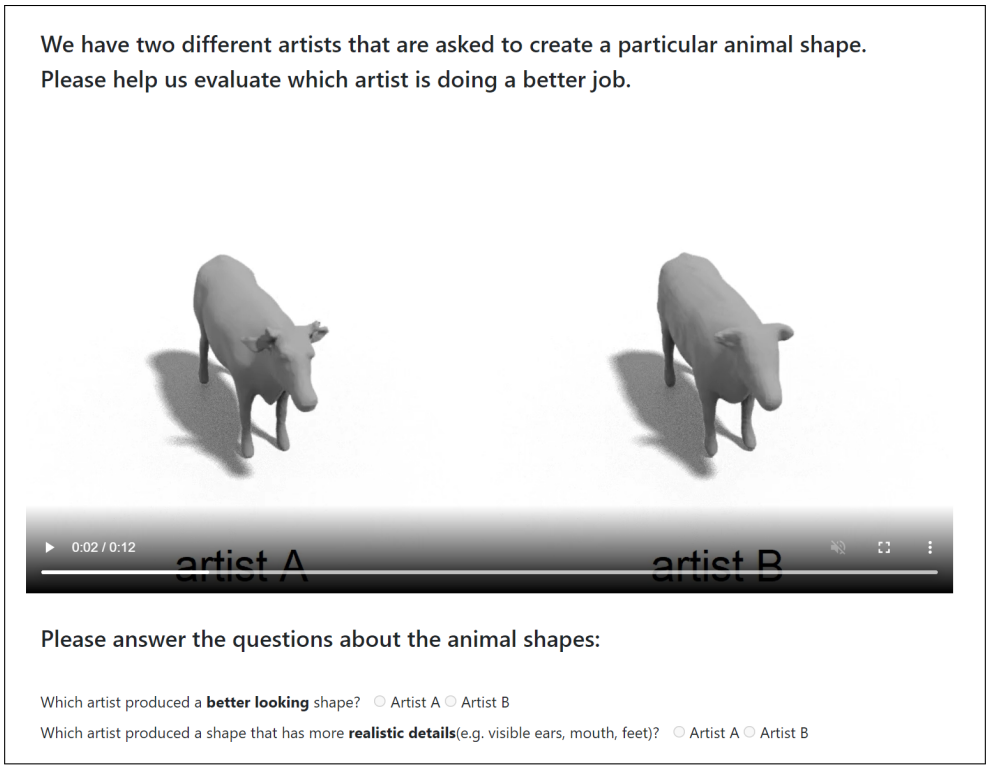

Figure K: The survey we presented to users on Amazon Mechanical Turk, where the two 3D animal shapes are displayed in a rotating view.

user allowed to work on our HITs has to have finished 1000 approved HITs on MTurk, with an approving rate higher than 95%.

We pay the participants of the user studies 0.02 US Dollars for each assignment. The average time required for one assignment is 9 seconds. Thus, the estimated hourly wage paid to the participants is $3600/9 \times 0.02 = 8$ US Dollars. The total amount spent on participant compensation is $5 \times 442 \times 3 \times 0.02 = 132.6$ US Dollars.

G Additional Discussions

Limitations and Future Work We discuss a few limitations to be solved in future work. First, our conditional generative model that predicts high resolution animals from coarse voxels fails to generate multi-modal shapes from a latent distribution. In the first to second rows of Fig. L, the similar heads of bears are generated for different inputs, not representing the diversity in natural 3D shapes. We would like to leverage more techniques from the Generative Adversarial Network literature to achieve multi-modal predictions, and allow further user control. Moreover, our model typically fails to generate large and irregular structures like antlers in the third row of Fig. L. We hypothesize the reason could be that the discriminator needs to see larger regions to realize it, and we plan to explore using multi-resolution discriminators in the future.

Second, although DMTET utilize volume subdivision to increase the resolution only around the surface, the encoder network still encodes features in a regular grid. Therefore, the resolution of the encoding is limited, leading to artifacts when reconstructing very high resolution details, as shown in Fig. M. To increase the resolution of the feature volume while maintaining low memory footprint, we would like to explore using octree-based feature volumes as in NGLOD [17].

Moreover, the effect of surface subdivision after applying volume subdivision is not obvious because the resolution of the generated surface is already very high. Ideally, low-curvature regions on the surface should be modeled only by a few control points and further converted to a parametric surface

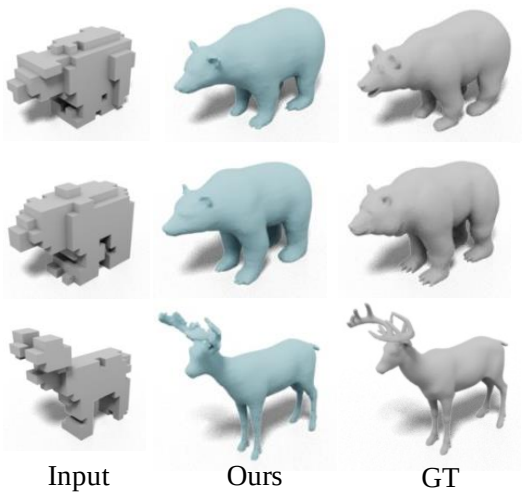

Figure L: **Failure cases for 3D shapes Synthesis from Coarse Voxels.** The generated details are similar for the first two examples regardless of the inputs. The antler in the third example looks unnatural.

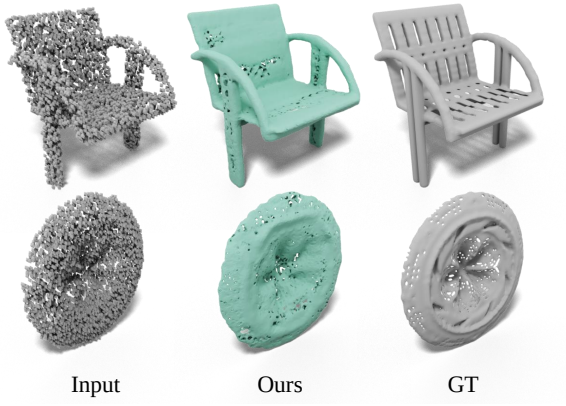

Figure M: **Failure cases for 3D Reconstruction from Point Clouds** Our model produces artifacts when reconstructing very high-resolution details.

by surface subdivision to reduce the computational cost. The volume subdivision should only be applied to regions that need more control points to model the underlying structure, such as high-frequency details. In the future, we would like to make volume subdivision focus on such regions to reduce the functional overlap between the two subdivision modules.

Lastly, the surface loss (e.g. Chamfer Distance) we adopt in our experiment is not monotonically decreasing. thus it suffers from bad local minimum. We find there is no guarantee that the shape converges to GT from a random initialization (even overfitting as shown in Fig. N), so we pair with L_{SDF} in learning to alleviate this issue. Without supervision or regularization on SDF, the model tends to produce double/broken surfaces, which is also reported in DMC [9]. In practice, when training a network for a learning task, we found a combination of L_{CD} with a less-weighted L_{SDF} gives the best result. Nevertheless, we would like to explore other surface losses in the future to relax the requirement on occupancy supervision. For example, we find the problem of local minimum is less a concern for image supervision (e.g. silhouette loss in MeshSDF's experiment setup).

Computational Costs All of our experiments and evaluations were run on our internal cluster. The network training in all experiments takes approximately 20,000 GPU hours on Nvidia V100 GPU.

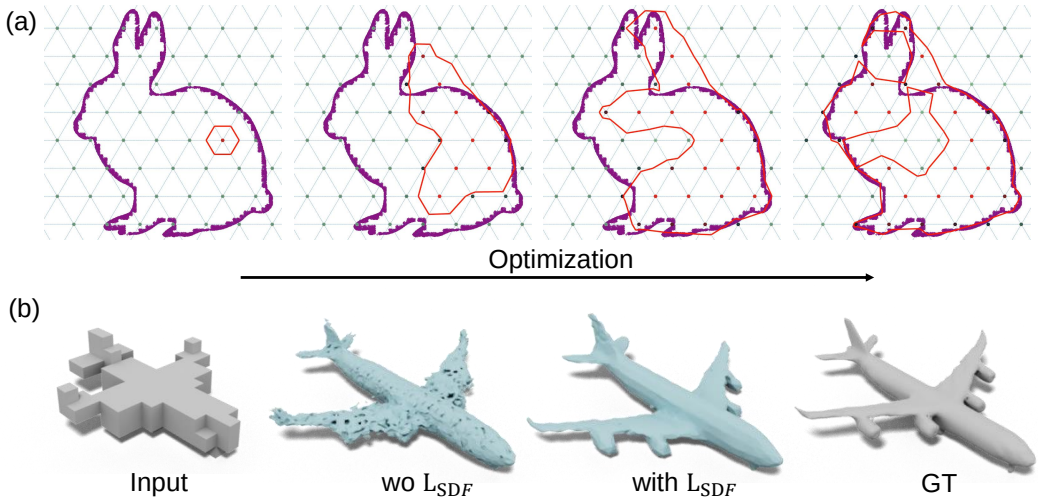

Figure N: **Illustration of local minimum of Chamfer Distance: a)** A 2D example of marching triangle. Here we optimize the surface parametrized by signed distance field (shown in red) to minimize the Chamfer Distance to the point cloud of a bunny (shown in purple). The optimization stuck at local minimum with wrong topology. **b)** Without L_{SDF} to regularize the learned SDF, our model tends to produce double/broken surfaces. Pairing L_{CD} with a less-weighted L_{SDF} generates a smooth surface.

References

[1] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pages 223–232. Wiley Online Library, 2003.

[2] Zhiqin Chen, Vladimir G. Kim, Matthew Fisher, Noam Aigerman, Hao Zhang, and Siddhartha Chaudhuri. Decor-gan: 3d shape detailization by conditional refinement. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[3] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[4] Akio Doi and Akio Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE TRANSACTIONS on Information and Systems*, 74(1):214–224, 1991.

[5] Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. In *Advances In Neural Information Processing Systems*, 2020.

[6] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9785–9795, 2019.

[7] Glen A Hansen, Rod W Douglass, and Andrew Zardecki. Mesh enhancement: Selected elliptic methods, foundations and applications, 2005.

[8] Krishna Murthy J., Edward Smith, Jean-Francois Lafleche, Clement Fuji Tsang, Artem Rozantsev, Wenzheng Chen, Tommy Xiang, Rev Lebaredian, and Sanja Fidler. Kaolin: A pytorch library for accelerating 3d deep learning research. *arXiv:1911.05063*, 2019.

[9] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, 2019.

[11] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, 2019.

[12] Charles Loop. Smooth subdivision surfaces based on triangles. January 1987.

[13] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[14] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[15] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020.

[16] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22468–22478. Curran Associates, Inc., 2020.

[17] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[18] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018.