HVPUNet: Hybrid-Voxel Point-cloud Upsampling Network

Juhyung Ha[1] Vibhas Kumar Vats[1] Soon-heung Jung[2] Alimoor Reza[3] David J. Crandall[1]

[1]Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN USA
[2]Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea
[3]Department of Mathematics and Computer Science, Drake University, Des Moines, IA USA

juhha@iu.edu, vkvats@iu.edu, zeroone@etri.re.kr, md.reza@drake.edu, djcran@iu.edu

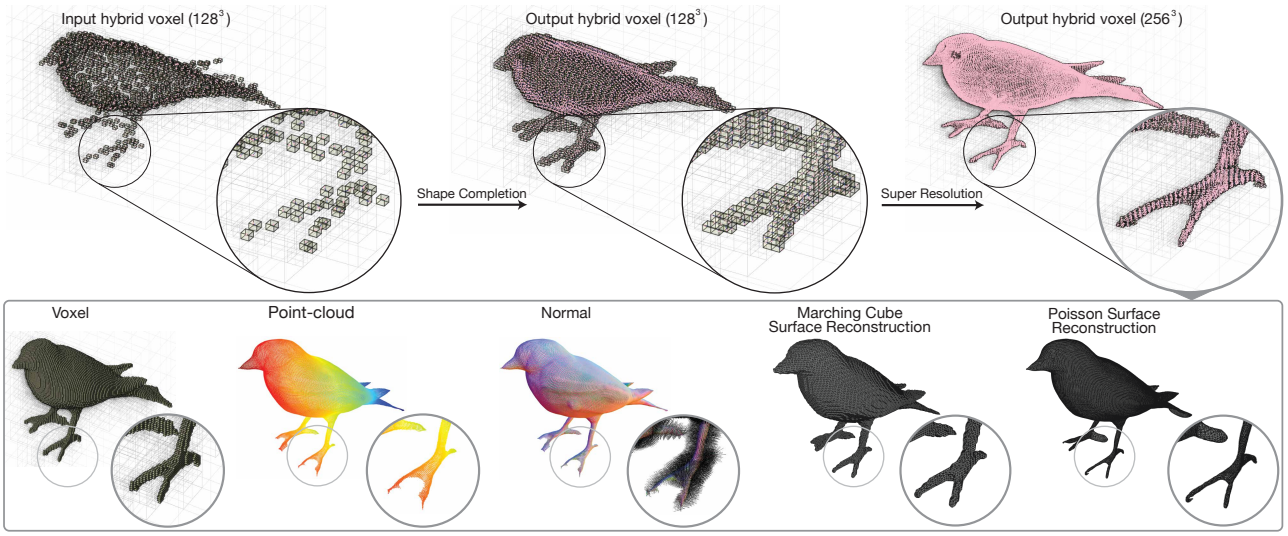

Figure 1. We propose an end-to-end technique, HVPUNet, for upsampling 3D point clouds. Unlike most existing techniques, it can upsample points at precise positions with lower computational cost by using hybrid voxels. Our hybrid voxel representation encodes continuous point locations in a structured voxel grid, allowing both precise and efficient reconstruction. HVPUNet consists of two parts: shape completion which imputes missing geometry in the sparse 3D input, and super-resolution which generates a more detailed reconstruction. The estimated hybrid voxel output with normal vectors can be used to reconstruct 3D surfaces.

Abstract

Point-cloud upsampling aims to generate dense point sets from sparse or incomplete 3D data. Most existing work uses a point-to-point framework. While this method achieves high geometric precision, it is slow because of irregular memory accesses to process unstructured point data. Alternatively, voxel-based methods offer computational efficiency by using regular grids, but struggle to preserve precise point locations due to discretization. To resolve this efficiency-precision trade-off, we introduce Hybrid Voxels, a representation that combines both voxel occupancy and a continuous point offset. We then present the Hybrid-Voxel Point-cloud Upsampling Network (HVPUNet), an efficient framework built upon this representation. HVPUNet integrates two key modules: (1) Shape Completion to restore missing geometry by filling empty voxels, and (2) Super-Resolution to enhance spatial resolution and capture finer surface details. We also use progressive refinement, operational voxel expansion, and implicit geometric learning. Experimental results demonstrate that HVPUNet can upsample point clouds at significantly lower computational cost than the state-of-the-art, but with comparable model accuracy.

1. Introduction

Spatial enhancement in 3D is a crucial yet challenging problem for applications including computer graphics, CAD, and virtual reality. Point-cloud upsampling aims to enhance the details of 3D models by generating denser points from

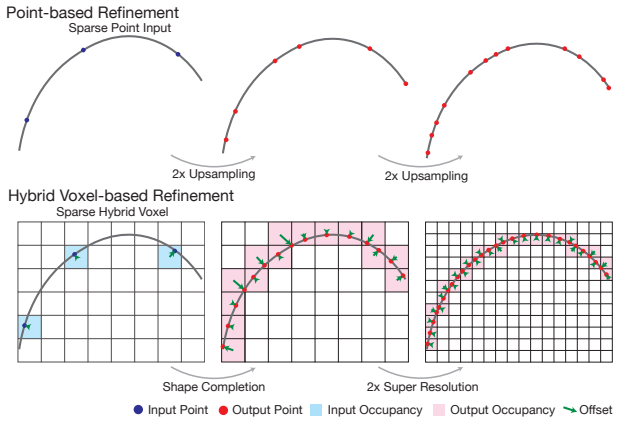

Figure 2. *Point-based* representations can be used to upsample point density for greater surface detail, but struggle with scalability in complex scenes. In contrast, our *Hybrid Voxel* representation combines discrete occupancy with continuous offset vectors, bridging structured voxel grids and continuous surfaces. Using Hybrid Voxels, we show how *Shape Completion* and *Super Resolution* can super-sample the spatial resolution of points.

noisy and sparse inputs. In the literature [17, 25, 35, 36, 39, 56], point-to-point (P2P) frameworks [9] are widely studied to enable precise 3D reconstruction in continuous space. However, they limit scalability and efficiency, as they rely on operations not optimized for large-scale data and can only generate a fixed number of points per inference regardless of the complexity of a given 3D model.

In contrast, voxel-based approaches [8, 44] offer efficiency and scalability for large models by using structured grids. These methods can dynamically adapt voxel occupancy based on the level of detail in a given model. However, quantizing continuous points into discretized voxel grids limits the ability to capture fine-grained details for high-fidelity reconstruction.

To address these limitations, we propose a Hybrid Voxel representation, combining the strengths of both paradigms. This representation encodes discrete occupancy within the voxel grid as well as offset vectors within each voxel to capture finer-grained positions in continuous space. Our approach enables better conversion between the voxel grid and point cloud, and supports efficient sparse 3D convolutions without sacrificing spatial precision. Figure 2 illustrates how a hybrid voxel representation can be converted to a point cloud and vice versa.

We introduce the Hybrid-Voxel Point-cloud Upsampling Network (HVPUNet), an end-to-end framework consisting of two modules: Shape Completion and Super-Resolution (see Figure 2). Shape Completion first densifies hybrid voxels and reconstructs geometry, and then Super-Resolution generates high-resolution hybrid voxels with enhanced surface detail. By using sparse 3D convolutions, HVPUNet efficiently processes large-scale 3D scenes, focusing computation only on active voxels thus reducing memory and computational costs. Additionally, our model can estimate surface normals which can be useful for surface reconstruction. To enhance scene reconstruction, we implement operational voxel expansion and early dilation [38] to broaden the receptive fields, thus enabling better imputation of neighboring voxels (see Figure 3).

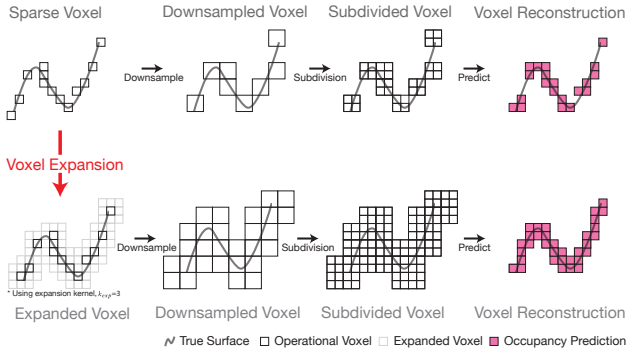

Figure 3. We expand the operational voxels around existing active voxels, leading to more complete scene reconstructions. The expanded voxels are unoccupied but operationally active.

Our network captures fine-grained geometric details not only by learning to predict binary occupancy, but also by estimating two continuous, implicit fields: unsigned distance fields (UDFs) and offset vector fields (OVFs). UDFs encode the shortest distance from each voxel to the nearest occupied voxel, while OVFs represent directional vectors from voxel centers to the closest surface. Unlike other voxel refinement methods that predict only binary occupancy [8, 23, 44, 51], these structural features enrich the network's understanding of implicit geometries, enabling more precise and complete reconstructions.

In summary, the key contributions of this work are:

- Hybrid Voxel Representation: Combines discrete occupancy and continuous offsets for more efficient and accurate upsampling;
- HVPUNet: A unified Shape Completion and Super-Resolution pipeline to generate high-resolution results with relatively low computational cost;
- Operational Voxel Expansion: Broadens the spatial context around each active voxel, encouraging more comprehensive scene coverage by allowing the network to access more candidate voxels to occupy; and
- Implicit Field Learning: UDFs and OVFs help improve reconstruction quality by allowing the model to capture implicit geometry.

2. Related Work

2.1. Point Cloud Upsampling

Point cloud upsampling aims to densify sparse 3D data by increasing the number of points. Most existing work on this problem follows a point-to-point (P2P) framework [9], where networks learn to extract features for each point by aggregating information from its local neighborhood or patch. Beginning with PU-Net [56], many approaches have used multi-layer perceptrons (MLPs), often enhanced by techniques such as adversarial training [1] (in PU-GAN [25]) and graph convolutions [15] (in PU-GCN [35]). Recent work [17, 36, 39] incorporates more advanced techniques like kernel point convolution [47], diffusion models [18], and gradient descent optimization [40]. RepKPU [39] introduces deformable kernel point representations, GradPU [17] optimizes learned distance fields via gradient descent, and PUDM [36] uses diffusion for flexible upsampling. While these methods can precisely reconstruct point-clouds in continuous space, they require more computation due to random memory access from irregular and unstructured 3D representation [27].

2.2. Voxel-based 3D Enhancement

Shape Completion aims to fill in missing voxels in 3D scenes. MinkowskiUNet [8] uses a U-Net in which downsampling determines receptive regions for completion. We extend this idea via operational voxel expansion to further enlarge the region in which new voxels may be filled. PVCNN [27] takes a hybrid approach by converting point clouds into voxels for 3D convolution and then reverting to points for upsampling. However, its final output is point-based, inheriting the same limitations of point clouds.

Voxel Super-Resolution [31, 48] typically uses dense 3D convolutions to enhance volumetric resolution, which is memory-intensive for large scenes. Sparse convolution [8, 23] can address this by computing only on active voxels, but generally cannot fill empty regions. To make super-resolution more feasible with sparse and incomplete data, we apply shape completion before voxel super-resolution.

2.3. Point-Voxel Hybrid Representation

Point-based methods offer precise localization but are less efficient; voxel-based methods are efficient but lack continuous coordinate support. Hybrid strategies have been used for feature extraction [7, 27, 45, 52, 53, 55]. Inspired by this, we define a hybrid voxel representation that combines occupancy and offset vectors in a unified structure for high-quality reconstruction.

2.4. Implicit Representation Learning

Recent 3D reconstruction research shows that learning *implicit representations* can significantly enhance surface reconstruction quality. Examples include Unsigned Distance Fields (UDFs) [6, 17], which are the distances to the nearest surfaces, Signed Distance Fields (SDFs) [14, 32], which is the same distance but negative inside the volume and postive outside, and Neural Vector Fields (NVFs) [5, 28, 34, 54, 59], which indicate directional offsets from voxel centers to the nearest surface. These implicit fields help models capture complex 3D geometry more precisely. We use UDFs and offset vector fields (OVFs) to encourage our model to learn implicit geometries, enabling more precise surface recovery and facilitating tasks like pruning (described below).

2.5. Sparse Convolution

In dense convolution, filter kernels are applied uniformly across the entire grid, wasting computation when large regions are empty. Sparse convolution operates only on operationally active voxels, reducing computational overhead. Frameworks like Minkowski Engine [8], TorchSparse++ [46], and SparseConvNet [13] have made these efficient and scalable. We use sparse convolution to allow processing large-scale scenes with our hybrid voxel approach.

3. Methods

We first introduce our Hybrid Voxel representation, and then present an overview of our Hybrid-Voxel Point-cloud Upsampling Network (HVPUNet).

3.1. Hybrid Voxels

To overcome limitations of point- or voxel-based approaches, we propose a hybrid voxel representation. Each hybrid voxel contains **Occupancy**, a binary flag indicating the presence of a point in the discrete voxel grid, and **Offset Vector** (v_{off}), a 3D vector from the voxel center to the nearest surface point. Formally, we define an offset vector,

$$v_{\text{off}} = p_{\text{surf}} - p_{\text{coord}}, \tag{1}$$

where p_{surf} is the nearest surface projection and p_{coord} is the center of the voxel coordinate. During inference, the surface point can be recovered,

$$\hat{p}_{\text{surf}} = \hat{p}_{\text{coord}} + \hat{v}_{\text{off}}. \tag{2}$$

In our implementation, we store each occupancy as a 1-channel sparse tensor and each offset as a 3-channel sparse tensor in TorchSparse++ [46], creating a 4-channel hybrid voxel. Figure 5 illustrates how attributes for hybrid voxels are obtained from continuous 3D surfaces.

3.2. HVPUNet

We designed HVPUNet to operate directly on our hybrid voxel representation. Its architecture is built for both geometric completion and detail enhancement, guided by implicit field learning and a progressive training strategy to improve computational efficiency.

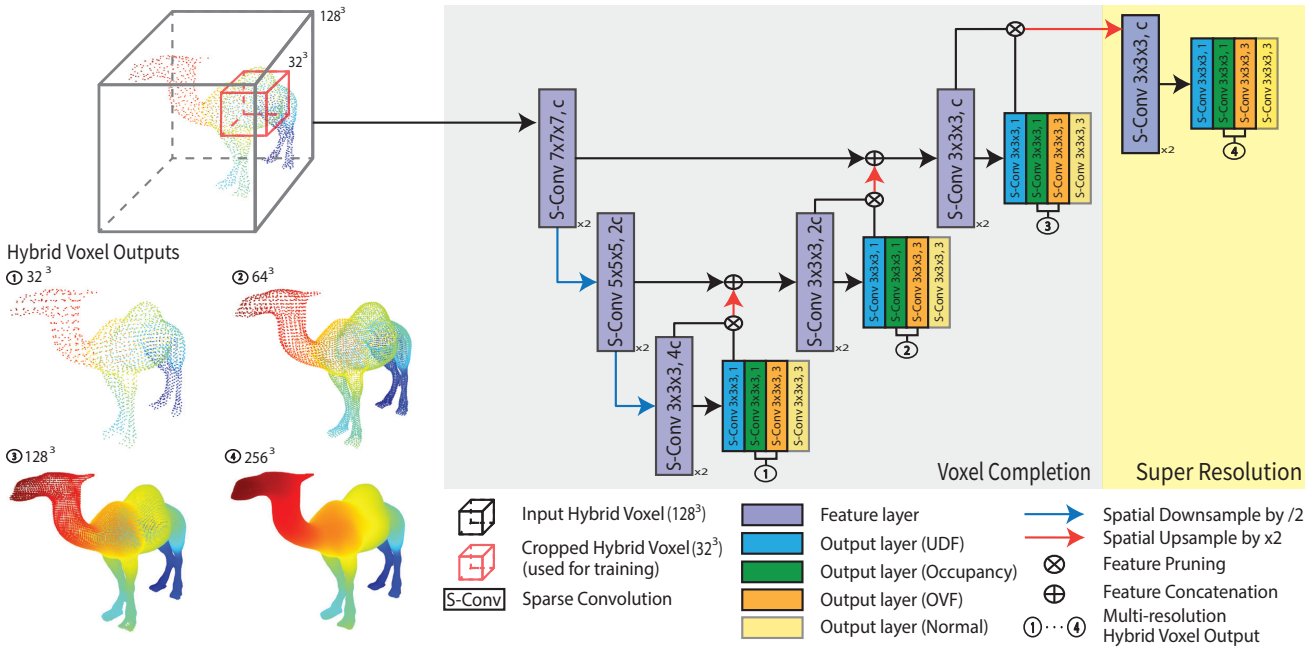

Figure 4. Overview of HVPUNet. An input point cloud is first converted to a 128^3 Hybrid Voxel representation. Shape Completion imputes missing points, and then Super-Resolution refines the reconstruction quality. With progressive training and refinement, the model produces outputs at multiple scales (32^3, 64^3, 128^3, and 256^3). At each scale, the network generates four outputs: unsigned distance field, occupancy, offset vector fields, and normal vectors. For memory-efficient mini-batch training, we randomly crop the input to 32^3 patches.

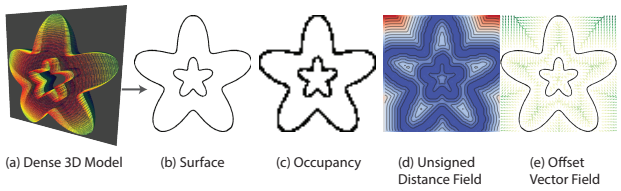

Figure 5. Illustration of unsigned distance fields (UDFs) and offset vector fields (OVFs), visualized in 2D for clarity. The point cloud in (a) and shape in (b) can be converted into the sampled occupancy grid in (c). The UDF in (d) represents the distance from each voxel to the closest occupied voxel, while the OVF in (e) consists of vectors from the center of each voxel to the closest surface.

Network Architecture. HVPUNet is a U-Net architecture with an extra upsampling module that uses sparse convolutions for computational efficiency. As shown in Figure 4, it consists of two main functional stages. First, *Shape Completion* processes the sparse input hybrid voxels and densifies the 3D shape by imputing missing geometry. It establishes a complete, yet coarse, representation of the object. Then, *Super-Resolution* progressively upscales the spatial resolution of the voxel grid, refining the coarse shape to reconstruct finer surface details.

The overall network takes 4-channel hybrid voxels (1-channel occupancy, 3-channel offset vector) as input. At multiple resolutions, it predicts 8-channel outputs for each voxel: 1-channel occupancy, a 3-channel offset vector field (OVF), a 3-channel normal vector, and a 1-channel unsigned distance field (UDF). The occupancy and OVF are used to reconstruct the final point cloud, while the normals can be used for surface reconstruction. The UDF and OVF serve as implicit learning targets, as explained next.

Implicit Geometry Learning via Distance Fields. To encourage our network to learn geometrically coherent structures, we train it not just to estimate binary occupancy but also two implicit fields that describe the underlying geometry: *Unsigned Distance Fields (UDFs)* and *Offset Vector Fields (OVFs)* (see Figure 5). *UDFs* encode the absolute distance from each voxel center to the nearest surface, which we compute for the ground truth using a Euclidean Distance Transform [10, 30]. By training the network to predict this field, we provide it with a smooth and continuous signal that describes the underlying surface geometry, which is essential for both robust learning and for our pruning strategy described below. *OVFs* contain, for each voxel, the 3D vector from its center to the closest point on the true surface. This directly provides sub-voxel precision, allowing us to recover fine-grained details that would be lost in a standard voxel grid. Unlike methods that require watertight meshes for signed distance fields (SDFs) [11, 33, 37, 50], our UDF/OVF approach works robustly with non-watertight and complex real-world models.

Progressive Refinement and Pruning. To manage the computational cost of high-resolution 3D processing, we use *progressive refinement* and *distance-based pruning*. First, the network is trained with multi-scale supervision to generate outputs at multiple resolutions, from coarse (32^3) to detailed (256^3). The lower-resolution stages learn to reconstruct overall shape, while the later, higher-resolution stages focus on refining high-frequency details. This progressive approach improves both learning efficiency and the final reconstruction quality. Within the refinement process, we use distance-based pruning during sparse convolution by removing any voxel whose predicted unsigned distance value is greater than a certain threshold. This helps to preserve a thin layer of voxels around surface boundaries, which provides context for more accurate convolution.

3.3. Operational Voxel Expansion

Sparse convolution only operates on non-empty voxels, so the number of occupied output voxels is limited by the number of non-empty input voxels. Generative sparse convolutions [8, 46] attempt to solve this by first downsampling the voxel grid. This operation can group a distant but occupied voxel into the same downsampled block as other nearby empty voxels. A subsequent generative transposed convolution can then fill in this local neighborhood during upsampling. However, this strategy is often insufficient when the input is highly sparse. If occupied voxels are separated by large empty spaces, they will remain isolated in separate blocks even after downsampling.

To address this, we introduce operational voxel expansion, which marks additional surrounding voxels as active with a pre-defined expansion kernel size ($k_{exp} = 7$), even if the voxels start out as empty. Figure 3 shows the impact of operational voxel expansion on the shape completion task compared to conventional generative sparse convolution [8]. Along with voxel expansion, we use early dilation [38] with larger kernel sizes ($k = 7$) at initial layers to further increase the receptive field.

3.4. Loss formulation

Our loss consists of four equally-weighted terms,

$$L_{\text{total}} = L_{\text{occ}} + L_{\text{udf}} + L_{\text{ovf}} + L_{\text{normal}}, \tag{3}$$

where each term is given by:

- **Occupancy Loss** (L_{occ}) is a Dice loss,

$$L_{\text{occ}} = 1 - \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i + \sum_i^N g_i + smooth}, \tag{4}$$

  where N is the number of active voxels, p_i is the occupancy prediction of the i-th voxel (ranging from 0 to 1 after a sigmoid), g_i is the ground truth occupancy of the i-th voxel, and *smooth* is a small constant (10^{-5}) to prevent division by zero.

- **UDF Loss** (L_{udf}) is an L1 loss that enforces accurate distance predictions,

$$L_{\text{udf}} = \frac{1}{N}\sum_i^N \left| y_{\text{udf}}^{(i)} - \hat{y}_{\text{udf}}^{(i)} \right|, \tag{5}$$

  where $y_{\text{udf}}^{(i)}$ and $\hat{y}_{\text{udf}}^{(i)}$ are the true and predicted scalar values for the absolute distance from each voxel i to the nearest occupied coordinate.

- **OVF Loss** (L_{ovf}) is an L1 loss for the 3D offset vectors,

$$L_{\text{ovf}} = \frac{1}{N}\sum_i^N \left| v_{\text{off}}^{(i)} - \hat{v}_{\text{off}}^{(i)} \right|, \tag{6}$$

  where $v_{\text{off}}^{(i)}$ and $\hat{v}_{\text{off}}^{(i)}$ are the true and predicted offset vectors for voxel i, defined in Section 3.1.

- **Normal Loss** (L_{normal}) is a combination of L1 and angular sine loss [2, 24] for normal estimation,

$$L_{\text{normal}} = L_{\text{sin}} + L_{L1}, \tag{7}$$

$$L_{\text{sin}} = \frac{1}{N}\sum_i^N \left| n^{(i)} \times \hat{n}^{(i)} \right|, \tag{8}$$

$$L_{\text{L1}} = \frac{1}{N}\sum_i^N \left| n^{(i)} - \hat{n}^{(i)} \right|, \tag{9}$$

  where n and $\hat{n}$ are the true and predicted normal vectors for voxel i.

By jointly optimizing these terms, we encourage both geometrically and orientationally accurate predictions.

3.5. Evaluation Methods

We use two types of metrics for evaluating our results compared to ground truth: **occupancy-based** and **point-based**.

Occupancy-based. Dice Coefficient and Intersection-over-Union (IoU) are used for occupancy-based metrics, which are computed within the discrete voxel space. For these metrics, both the ground truth and the output point clouds are voxelized into an isotropic grid at a resolution of 128^3, and evaluated using voxel-wise classification metrics. These metrics assess the completeness of the predicted points by measuring how well they cover the surfaces.

Point-based. We use point-to-face (P2F) and face-to-point (F2P) distances [20] for point-based metrics,

$$\text{P2F}_{\text{avg}} = \frac{1}{|\hat{P}|}\sum_{\hat{p}\in\hat{P}} \min_{f\in\mathcal{F}}\big(\text{dist}(\hat{p}, f)\big), \tag{10}$$

$$\text{P2F}_{\text{max}} = \max_{\hat{p}\in\hat{P}}\Big(\min_{f\in\mathcal{F}}\big(\text{dist}(\hat{p}, f)\big)\Big), \tag{11}$$

$$\text{F2P}_{\text{avg}} = \frac{1}{|F|}\sum_{f\in F} \min_{\hat{p}\in\hat{P}}\big(\text{dist}(\hat{p}, f)\big), \tag{12}$$

$$\text{F2P}_{\text{max}} = \max_{f\in F}\Big(\min_{\hat{p}\in\hat{P}}\big(\text{dist}(\hat{p}, f)\big)\Big), \tag{13}$$

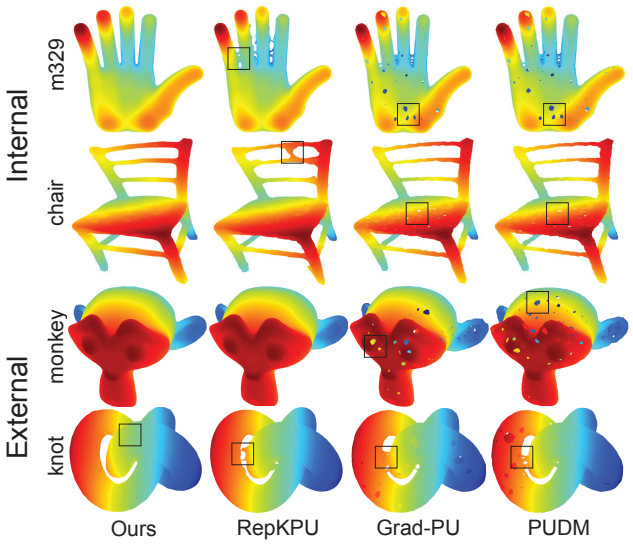

Figure 6. Sample model outputs from PU-GAN's test data (*Internal*) and from an *External* source. These samples reflect qualitative visualizations of shaded results from Table 1. Bounding boxes highlight artifacts. Zoom in for details.

where

$$\text{dist}(\hat{p}, f) = \min_{p \in f} \|\hat{p} - p\|. \tag{14}$$

$\hat{P}$ and F are the sets of predicted points and surfaces, $\hat{p}$ denotes a predicted point, f is a face in the ground truth mesh, and p is the closest point on f to $\hat{p}$.

P2F quantifies the distance from each predicted point to the nearest ground truth surface, while F2P measures the distance from the ground truth surface to the closest predicted point. These metrics are reported as both average and maximum values, to capture both typical accuracy across all points as well as outliers. We avoid direct point cloud comparison such as Chamfer [9] and Hausdorff distances [3] in our study due to their reliance on sampled ground truth points, which only partially represent true 3D surfaces [56].

4. Results

4.1. Experimental Setup

Data. We use the **PU-GAN** [25] and **PU1K** datasets [35], which include 147 3D models (120 training, 27 testing) and 1,147 3D models (1,020 training, 127 testing), respectively, all in a mesh format. We prepare multi-resolution target variables from 32^3 to 256^3 (see Figure 4). Target variables are converted from meshes as follows: (1) voxelize the mesh into discrete voxels, (2) generate offset vector fields by calculating the directional vector between each voxel center and closest surface, (3) map normal for each

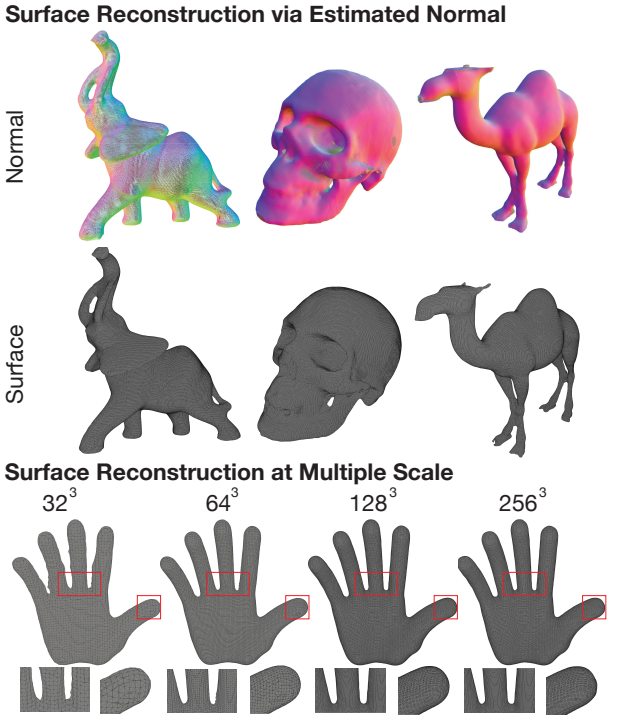

Figure 7. Normal estimation and surface reconstruction. HVPUNet estimates normal vectors, which can be used for surface reconstruction. We use Poisson Surface Reconstruction [21] on the estimated normals.

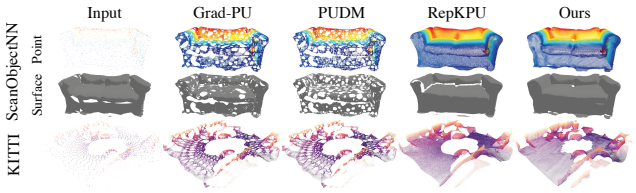

Figure 8. Sample outputs from real-world 3D scanned datasets (ScanObjectNN [49] and KITTI [12]). Please zoom in for details.

hybrid voxel to the normal from the closest surface, and (4) calculate unsigned distance fields with a Euclidean distance transform [30] on the voxelized grid. Overall, the target variables have 8 channels of information: 1 for occupancy, 3 for offset vector, 3 for normal, and 1 for unsigned distance. The occupancy and offset vector are used to reconstruct final point-cloud, unsigned distance is used in pruning, and estimated normal is used for surface reconstruction. For the input, we follow other point-cloud upsampling methods and sample 2048 points from the mesh. We then convert each point to a hybrid voxel in 128^3 resolution to feed into our model. Each input hybrid voxel has 4 channels (1 for occupancy and 3 for offset vector).

Implementation Details. In training, we use mini-batches of size 8, each containing 32^3 randomly cropped hybrid

voxels from the 128^3. We train our model in a progressive manner, first training lower-resolution modules and then incrementally adding higher-resolution modules. In total, we train our model with 15 epochs for each lower resolution (32^3, 64^3, 128^3) and 155 epochs at the highest resolution (256^3). Additionally, we use a voxel expansion kernel size (k_{exp} in Section 3.3) of 7, and a pruning distance threshold of 3. For hardware, we used 2 AMD EPYC 74F3 processors (CPU) and 1 RTX L40s (GPU). Training required approximately 20GB of VRAM, while inference VRAM consumption is reported in Table 1.

4.2. Results

Results on PU-GAN. Tables 1 and 2 compare results of our method with other state-of-the-art point cloud upsamplers including Grad-PU [17], PUDM [36], and RepKPU [39]. Table 1 shows results at various scaling factors. Since we use a different upscaling strategy than other point cloud upsamplers, we selected the highest upsampling factor for comparison. Occupancy-based metrics (Dice and IoU) provide insight for scene completion, and our method outperforms the other models. Point-based metrics show mixed performance, indicating the precision varies by methods. Even though other point-cloud upsamplers are directly trained on regression for precise point localization, our model shows best results on maximum scores in point-based metrics, indicating better robustness to outliers [29, 41]. Furthermore, Table 2 demonstrates that our model performs robustly under diverse conditions, including corrupted inputs (random, noisy, or undersampled) and arbitrary upsampling factor.

We report model efficiency in Table 1, including model size, average inference VRAM consumption, and average runtime. Our method is the second lightest in terms of model size, but dramatically outperforms others in memory efficiency and inference speed.

Results on PU1K. Table 3 shows quantitative results on PU1K [35]. On this dataset, our model has superior performance on Dice, IoU, and F2P while other metrics show mixed rankings. We used an upsampling factor of 256x and 256^3 for point-based and our method, respectively.

Qualitative Results. Sample visualizations are shown in Figure 6 using test cases from PU-GAN [25] as well as external sources (Open3D [58] and Sketchfab [22]). We show samples at the highest scaling factors for each method (x256 for point-cloud upsamplers and 256^3 for our method). The visualization indicates that Grad-PU [17] and PUDM [36] suffer from incomplete scene reconstruction (holes). RepKPU [39] and our method show more complete scene coverage, although RepKPU can fill points in incorrect regions. Additionally, Figure 8 shows sample visualizations of real-world 3D scanned datasets, including ScanObjectNN [49] and KITTI [12]. Overall, our method shows more complete and precise reconstruction of surface boundaries.

4.3. Ablation Study

Table 4 shows the results of an ablation study to investigate the effect of each component of our model. Both voxel expansion and distance-based pruning help in accuracy; the performance gain is especially dramatic in occupancy-based metrics. Furthermore, operational voxel expansion starts producing comparable performance from threshold of 5, while optimal is 7 as in our final model.

4.4. Normal Estimation

While the primary objective of our work is point-cloud upsampling, we also explored normal estimation to demonstrate how our framework can support downstream tasks like surface reconstruction. Figure 7 shows predicted normals and their utility in surface reconstruction [4, 21]. This illustration highlights that our method not only enhances point density but can also provide orientation cues for more coherent and visually pleasing surfaces.

5. Limitations and Future Work

Unlike P2P upsampling frameworks, our approach follows a different path using hybrid voxel and other attributes, providing a unique solution. Since this unique problem setting allows dynamic point-cloud upsampling factor, it is hard to compare with other fixed-scaled upsamplers since some evaluation metrics can be sensitive to the number of points. Furthermore, we provide normal estimation performance only in a qualitative manner. In future work, we plan to focus more on Point-to-Surface reconstruction [16, 19, 57] by expanding our work with learnable surface extractors [26, 42, 43].

6. Conclusion

We introduced HVPUNet, a unified method that combines point- and voxel-based strategies for computationally efficient point-cloud upsampling. By fusing discrete occupancy and continuous offset vectors, our hybrid representation preserves fine geometry while benefiting from the computational efficiency of sparse 3D convolutions. Experiments show that HVPUNet produces structurally complete and detailed reconstructions at low computational cost.

7. Acknowledgments

This work was supported by an Electronics and Telecommunications Research Institute (ETRI) grant funded by the government of the Republic of Korea (25ZC1110, The research of the basic media-contents technologies).

Method	Scale	# points (10^3)	Occupancy ↑		Point ↓				Complexity		
					P2F ($\times 10^3$)		F2P ($\times 10^3$)				
			Dice	IoU	Avg	Max	Avg	Max	# param. (10^6) ↓	VRAM (GB) ↓	Runtime (sec) ↓
Grad-PU [17]	16x	32.768	0.591	0.425	1.949	27.923	1.679	22.467	**0.067**	1.082	2.095
	64x	131.072	0.677	0.516	2.059	28.102	1.273	22.080		4.329	28.868
	256x	524.288	0.698	0.539	**2.143**	28.409	1.127	22.027		17.315	576.160
PUDM [36]	16x	32.768	0.517	0.355	3.205	29.131	2.530	25.324	11.544	1.164	4.269
	64x	131.072	0.601	0.452	3.343	30.001	2.018	25.156		4.511	22.148
	256x	524.288	0.639	0.461	3.425	30.365	1.802	24.949		17.899	272.806
RepKPU [39]	16x	32.768	0.581	0.417	2.061	43.613	1.607	16.678	2.048	2.116	1.273
	64x	131.072	0.674	0.514	2.167	46.247	1.122	16.234		8.417	18.218
	256x	524.288	0.695	0.539	2.261	46.822	**0.963**	16.238		33.621	345.387
Ours	128^3	99.585	0.610	0.443	1.783	25.556	1.345	14.443	0.449	0.868	0.134
	256^3	416.131	**0.707**	**0.549**	2.736	**26.023**	1.756	**15.723**		**2.817**	**1.543**

Table 1. Results on the PU-GAN dataset [25], using two types of metrics: (1) Occupancy-based using Dice and IoU, and (2) Point-based using Point-to-Face (P2F) and Face-to-Point (F2P). We also show complexity analysis including (1) model size, (2) inference VRAM consumption, and (3) inference runtime. The best result is in **bold** and second-best in underline. Point-cloud upsampler methods upscale by increasing the number of points, while our method uses resolution-based upscaling.

Experiment	Method	Dice	IoU	P2F (Avg)	P2F (Max)	F2P (Avg)	F2P (Max)
Random	Grad-PU	0.591	0.428	**2.295**	38.857	2.688	40.499
	PUDM	0.511	0.404	4.435	74.031	4.457	77.579
	RepKPU	0.659	0.509	2.465	50.870	**1.717**	25.260
	Ours	**0.670**	**0.517**	2.792	**33.435**	1.895	**24.120**
Noisy ($\tau = 0.01$)	Grad-PU	0.457	0.304	5.386	42.313	2.311	28.321
	PUDM	0.476	0.32	5.126	41.200	2.375	29.393
	RepKPU	0.509	0.348	6.017	65.168	**1.669**	**20.242**
	Ours	**0.514**	**0.355**	**5.098**	**42.595**	1.852	20.378
Under (512 points.)	Grad-PU	0.516	0.355	3.731	39.164	4.206	57.109
	PUDM	0.465	0.310	5.043	41.656	4.658	56.155
	RepKPU	0.550	0.393	3.421	62.195	2.068	38.911
	Ours	**0.604**	**0.445**	**2.783**	**37.335**	**1.891**	**36.940**
Arbitrary ($\times 50$)	Grad-PU	**0.667**	**0.512**	2.106	29.122	1.369	22.476
	PUDM	0.592	0.428	3.272	28.995	2.085	25.451
	RepKPU	0.652	0.490	**1.947**	47.964	**1.139**	17.618
	Ours	0.666	**0.512**	2.830	**26.699**	1.938	**15.888**

Table 2. Performance under (1) corrupted inputs (*random*, *noisy*, or *undersampled*) and (2) *arbitrary* upsampling. For *undersampled* inputs, we used 512 input points, compared to the 2,048 points used in Table 1. For *arbitrary* upsampling, we used farthest point sampling, following RepKPU [39] and Grad-PU [17].

Method	Occupancy ↑		Point ↓			
			P2F ($\times 10^3$)		F2P ($\times 10^3$)	
	Dice	IoU	Avg	Max	Avg	Max
Grad-PU	0.526	0.366	1.388	16.423	1.612	13.873
PUDM	0.414	0.305	2.891	18.413	3.397	18.890
RepKPU	0.485	0.339	**0.880**	**13.872**	2.039	14.186
Ours	**0.537**	**0.376**	1.507	25.438	**1.262**	**10.101**

Table 3. Our reconstruction results on the PU1K dataset [35] compared to Grad-PU [17], PUDM [36], and RepKPU [39]. We use a scaling factor of 16 for point cloud-based methods and a target resolution of 128^3 for ours.

Method	Occupancy ↑		Point ↓			
			P2F ($\times 10^3$)		F2P ($\times 10^3$)	
	Dice	IoU	Avg	Max	Avg	Max
no-exp	0.440	0.283	20.32	71.95	2.105	22.189
exp-3	0.481	0.336	13.332	63.007	1.811	9.122
exp-5	0.609	0.442	8.667	60.067	1.420	**8.536**
exp-9	0.602	0.437	2.201	30.527	1.061	8.896
prune-occ	0.458	0.298	1.899	35.424	**0.938**	9.643
Final	**0.610**	**0.443**	**1.783**	**25.556**	1.345	14.443

Table 4. Ablation study results on the PU-GAN dataset with 128^3 spatial resolution. **no-exp** represents no voxel expansion, **exp-i** represents voxel expansion size as i, and **prune-occ** represents training without L_{udf} and using occupancy probability for the pruning criteria instead of UDFs.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *ICML*, pages 40–49. PMLR, 2018. 3

[2] Yizhak Ben-Shabat and Stephen Gould. Deepfit: 3d surface fitting via neural network weighted least squares. In *ECCV*, pages 20–34, Cham, 2020. Springer International Publishing. 5

[3] Matthew Berger, Joshua A. Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T. Silva. A benchmark for surface reconstruction. *ACM Trans. Graph.*, 32(2), 2013. 6

[4] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999. 7

[5] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, pages 5932–5941, 2019. 3

[6] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *NeurIPS*, 2020. 3

[7] Jaesung Choe et al. Deep point cloud reconstruction. In *ICLR*, 2022. 3

[8] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, pages 3070–3079, 2019. 2, 3, 5

[9] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, pages 605–613, 2017. 2, 3, 6

[10] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 8 (19):415–428, 2012. 4

[11] Nicole Feng and Keenan Crane. A heat method for generalized signed distance. *ACM Trans. Graph.*, 43(4), 2024. 4

[12] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013. 6, 7

[13] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018. 3

[14] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *ICML*, pages 3789–3799. PMLR, 2020. 3

[15] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. 3

[16] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *ACM Trans. Graph.*, 39(4), 2020. 7

[17] Yun He, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. Grad-pu: Arbitrary-scale point cloud upsampling via gradient descent with learned distance functions. In *CVPR*, 2023. 2, 3, 7, 8

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, Red Hook, NY, USA, 2020. Curran Associates Inc. 3

[19] ZhangJin Huang, Yuxin Wen, ZiHao Wang, Jinjuan Ren, and Kui Jia. Surface reconstruction from point clouds: A survey and a benchmark. *TPAMI*, 46(12):9727–9748, 2024. 7

[20] Justin Johnson, Nikhila Ravi, Jeremy Reizenstein, David Novotny, Shubham Tulsiani, Christoph Lassner, and Steve Branson. Accelerating 3d deep learning with pytorch3d. In *SIGGRAPH Asia 2020 Courses*, New York, NY, USA, 2020. ACM. 5

[21] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, page 61–70, Goslar, DEU, 2006. Eurographics Association. 6, 7

[22] kluchek. Hd skull from ct scan. Sketchfab, 2019. Accessed: July 30, 2025. URL: https://sketchfab.com/3d-models/hd-skull-from-ct-scan-069dee7fd56b48688bb7e066c21d3a40. 7

[23] Jianning Li, Stefan Scherer, and Jan Egger. Sparse convolutional neural networks for efficient skull reconstruction. *Scientific Reports*, 13(20420), 2023. 2, 3

[24] Qing Li, Yu-Shen Liu, Jin-San Cheng, Cheng Wang, Yi Fang, and Zhizhong Han. HSurf-Net: Normal estimation for 3D point clouds by learning hyper surfaces. In *NeurIPS*, pages 4218–4230. Curran Associates, Inc., 2022. 5

[25] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: a point cloud upsampling adversarial network. In *ICCV*, 2019. 2, 3, 6, 7, 8

[26] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *CVPR*, pages 2916–2925, 2018. 7

[27] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *NeurIPS*, 2019. 3

[28] Andrew Luo, Tianqin Li, Wen-Hao Zhang, and Tai Sing Lee. Surfgen: Adversarial 3d shape synthesis with explicit surface discriminators. In *ICCV*, pages 16238–16248, 2021. 3

[29] Baraka Jacob Maiseli. Hausdorff distance with outliers and noise resilience capabilities. *SN Computer Science*, 2(5):358, 2021. 7

[30] C.R. Maurer, Rensheng Qi, and V. Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *TPAMI*, 25(2): 265–270, 2003. 4, 6

[31] Priyanka Nandal, Sudesh Pahal, Ashish Khanna, and Placido Rogério Pinheiro. Super-resolution of medical images using real esrgan. *IEEE Access*, 12:176155–176170, 2024. 3

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 3

[33] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 4

[34] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 3

[35] Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. Pu-gcn: Point cloud upsampling using graph convolutional networks. In *CVPR*, pages 11683–11692, 2021. 2, 3, 6, 7, 8

[36] Wentao Qu, Yuantian Shao, Lingwu Meng, Xiaoshui Huang, and Liang Xiao. A conditional denoising diffusion probabilistic model for point cloud upsampling. In *CVPR*, pages 20786–20795, 2024. 2, 3, 7, 8

[37] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. In *NeurIPS*, pages 22468–22478. Curran Associates, Inc., 2020. 4

[38] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *CVPR*, 2024. 2, 5

[39] Yi Rong, Haoran Zhou, Kang Xia, Cheng Mei, Jiahao Wang, and Tong Lu. Repkpu: Point cloud upsampling with kernel point representation and deformation. In *CVPR*, pages 21050–21060, 2024. 2, 3, 7, 8

[40] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv 1609.04747*, 2017. 3

[41] Michael D. Shapiro and Matthew B. Blaschko. On hausdorff distance measures. Technical report, University of Mas-

sachusetts Amherst, Computer Science Department, 2004. 7

[42] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *NeurIPS*, 2021. 7

[43] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Trans. Graph.*, 42(4), 2023. 7

[44] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, pages 190–198, 2017. 2

[45] Haotian* Tang, Zhijian* Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *ECCV*, 2020. 3

[46] Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han. Torchsparse++: Efficient training and inference framework for sparse convolution on gpus. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2023. 3, 5

[47] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *ICCV*, 2019. 3

[48] Ryosuke Ueda and Yuta Muraki. Super-resolution of voxel model using 3D ESRGAN. In *International Workshop on Advanced Imaging Technology (IWAIT) 2024*, page 131641E. SPIE, 2024. 3

[49] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019. 6, 7

[50] Ruian Wang, Zixiong Wang, Yunxiao Zhang, Shuangmin Chen, Shiqing Xin, Changhe Tu, and Wenping Wang. Aligning gradient and hessian for neural signed distance function. In *NeurIPS*, 2023. 4

[51] Xiaogang Wang, Marcelo H Ang, and Gim Hee Lee. Voxel-based network for shape completion by leveraging edge generation. In *ICCV*, pages 13169–13178, 2021. 2

[52] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2dpass: 2d priors assisted semantic segmentation on lidar point clouds. In *ECCV*, pages 677–695. Springer, 2022. 3

[53] Xu Yan, Chaoda Zheng, Zhen Li, Shuguang Cui, and Dengxin Dai. Benchmarking the robustness of lidar semantic segmentation models. *arXiv preprint arXiv:2301.00970*, 2023. 3

[54] Xianghui Yang, Guosheng Lin, Zhenghao Chen, and Luping Zhou. Neural vector fields: Implicit representation by explicit learning. In *CVPR*, pages 16727–16738, 2023. 3

[55] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *CVPR*, pages 1628–1637, 2020. 3

[56] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *CVPR*, 2018. 2, 3, 6

[57] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *NeurIPS*, 2022. 7

[58] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 7

[59] Xiangyu Zhu et al. Nerve: Neural volumetric edges for parametric curve extraction from point cloud. In *CVPR*, 2023. 3