Deep Surface Reconstruction from Point Clouds with Visibility Information

Raphael Sulzer, Loic Landrieu, Alexandre Boulch, Renaud Marlet, Bruno Vallet

Deep Surface Reconstruction from Point Clouds with Visibility Information

Raphael Sulzer*, Loïc Landrieu*, Alexandre Boulch‡, Renaud Marlet†‡ and Bruno Vallet*
*LASTIG, Univ Gustave Eiffel, IGN-ENSG, F-94160 Saint-Mandé, France
Email: firstname.lastname@ign.fr
†LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France
‡Valeo.ai, Paris, France

***Abstract*—Most current neural networks for reconstructing surfaces from point clouds ignore sensor poses and only operate on raw point locations. Sensor visibility, however, holds meaningful information regarding space occupancy and surface orientation. In this paper, we present two simple ways to augment raw point clouds with visibility information, so it can directly be leveraged by surface reconstruction networks with minimal adaptation. Our proposed modifications consistently improve the accuracy of generated surfaces as well as the generalization ability of the networks to unseen shape domains.**

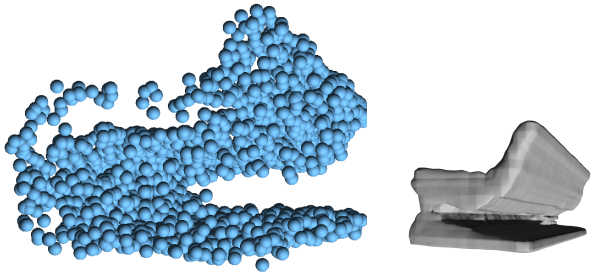
I. Introduction

The problem of reconstructing a watertight surface from a point cloud has recently been addressed by a variety of deep learning based methods. Compared to traditional approaches, deep surface reconstruction (DSR) can learn shape priors [1], [2] and leverage shape similarities [3] to complete missing parts [4], filter outliers, or smoothen noise in defect-laden point clouds. DSR methods, however, often derive priors from training datasets with few shape classes, generalizing poorly to unseen categories or datasets. Learning more local priors improves consistency across different objects or scenes [5], [6] but may result in higher sensitivity to noise or other defects. Besides, lack of global context complicates surface orientation.

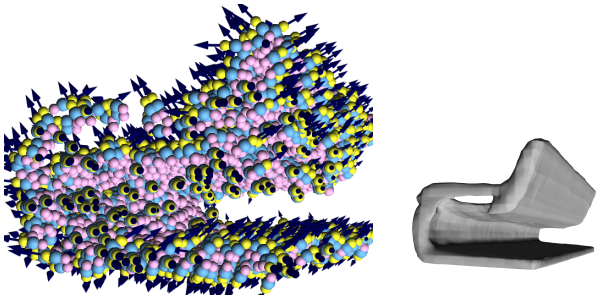
For real world point clouds, usually acquired via active or passive methods such as LiDAR scanning or multi-view stereo (MVS), the sensor position can be known and used to relate each observed point with a line of sight. Such visibility information can then help to orient surface normals [7] or predict occupancy [8], [9], [10]. While visibility is key for MVS, it has largely been ignored by DSR methods. In fact, sensor positions are usually not given in reconstruction benchmarks from point clouds. To remedy this, we consider virtual scanning rather than uniform sampling, and we show that many DSR methods can easily be adapted to benefit from visibility (cf. Figure 1). Our main contributions are as follows:

- We propose two simple ways to add visibility information to 3D point clouds, and we detail how to adapt DSR methods to utilize them, with very little changes.
- Using synthetic and real data, at object and scene level, we show for a wide range of state-of-the-art DSR methods that models leveraging visibility reconstruct higher-quality surfaces and are more robust to domain shifts.

Incidentally, our benchmarks also allow us to compare a range of recent state-of-the-art DSR methods on the same ground.



(a) Reconstruction using only the points position.



(b) Reconstruction with visibility augmented point cloud.

Fig. 1. **Surface Reconstruction with Visibility Information.** We augment each 3D point ● with a sightline vector → pointing towards the sensor observing it. Additionally, two auxiliary points are placed before ● and after ● the observed point along the sightline. This allows DSR networks, with very little modification, to reconstruct a significantly more accurate surface.

II. Related Work

Many traditional surface reconstruction methods use visibility information [8], [9], [10], [11], [12], [13], [14]. They are usually based on a 3D Delaunay tetrahedralization, which is intersected with lines of sight to attribute visibility features to Delaunay cells. While such methods can scale to billions of points [15] and are robust to moderate levels of noise and outliers, they do not incorporate learned shape priors.

In contrast, recent DSR methods have shown to produce more accurate surfaces than traditional approaches, especially for shape categories encountered during training. Many DSR methods use an implicit surface representation, either based on occupancy [2], [16], [17], or on the distance to the surface, whether it is signed [1], [17], [18], [19] or unsigned [20], [21], [22], [23]. To integrate local information, different forms

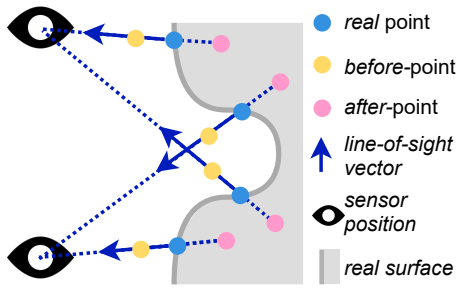


Fig. 2. **Visibility-Augmented Point Cloud.** Each observed point is associated to a sightline unit vector pointing towards its sensor. Two new points before and after each point are added. They help to disambiguate occupancy.

of convolutions are used, either on regular grids [4], [5], [24], [25], [26], directly on points [27], [28] or via an MLP instead [29]. Other methods rather use an explicit surface representation such as a mesh, which is deformed [3] or whose elements classified [6], [30].

A key issue is to get a sense of surface point orientation, to choose between reconstructing a thin volume (two main opposite orientations) or a thicker one (one main orientation at void-matter interface). Some methods dismiss the orientation issue by requiring oriented normals as input [5], [27], [31], [32], albeit producing such normals is a challenging task in itself [7], [33], [34]. We show that oriented normals can be advantageously replaced by visibility information.

Only few deep-learning methods make use of visibility information, typically from multiple views with camera pose information. RayNet [35] aggregates features from pixels of different views that intersect in the same voxel, but it outputs a dense point cloud, not a watertight surface mesh. Neural radiance fields [36], [37] somehow also model the free space between a point and its sensor. They, however, generally assume numerous and dense views (*i.e.*, images), and leverage little or no shape priors. We argue that DGNN [6], that classifies Delaunay cells with a graph neural network, currently is the only general DSR method from point clouds with visibility. However, DGNN relies on handcrafted visibility features requiring substantial geometry processing, while, we propose to directly augment the input point clouds. For point clouds for which visibility information is not available, Vis2Mesh [38] shows that rendering virtual views and learning point sensor visibility can significantly improve the reconstruction quality of a traditional method.

III. Method

We consider a 3D point cloud P where each point $p \in P$ has some coordinates $X_p \in \mathbb{R}^3$ and knows the position $S_p \in \mathbb{R}^3$ of a sensor observing it. Instead of only using the raw point coordinates $(X_p)_{p\in P}$ as the input $(I_p)_{p\in P}$ of a DSR network, we propose two simple ways to augment point cloud P with visibility information, and adapt DSR methods accordingly.

A. Sightline Vector (SV)

For each $p \in P$, we define a unit vector $\mathbf{v}_p$ pointing from the observation X_p to the sensor S_p: $\mathbf{v}_p = (S_p - X_p)/\|S_p - X_p\|$. This contains useful information for surface orientation. We normalize the vector as the distance to the sensor is not as relevant as the viewing angle.

B. Auxiliary Points (AP)

To help the network predict empty and full space immediately in front of and behind the observed surface, we consider two auxiliary points to each point p: a *before-point* p_b and an *after-point* p_a, located along the sightline on each side of p: $X_{p_\mathrm{b}} = X_p + d\mathbf{v}_p$, $X_{p_\mathrm{a}} = X_p - d\mathbf{v}_p$, where d is a characteristic distance in the point cloud P, e.g., the average distance from a point to its nearest neighbor. By construction, p_b is likely outside the scanned object or scene (modulo sensing noise and outliers), and p_a, likely inside (modulo object thickness too).

C. Visibility-Augmented Point Cloud

We use sightline vectors and auxiliary points to add visibility information to an input point cloud, separately or together.

(SV) To use sightline information only, we simply concatenate the sightline vector channelwise to the point coordinates to form the network input: $I_p = (X_p \oplus \mathbf{v}_p) \in \mathbb{R}^6$.

(AP) To use auxiliary points only, we add before-points p_b and after-points p_a to P, with tags $\mathbf{t} \in \mathbb{R}^2$ concatenated to point coordinates to identify the point type, *i.e.*, $I_q = (X_q \oplus \mathbf{t}_q) \in \mathbb{R}^5$ with $q \in \{p, p_\mathrm{b}, p_\mathrm{a}\}$, where $\mathbf{t}_p = [0\ 0]$ (observed point), $\mathbf{t}_{p_\mathrm{b}} = [1\ 0]$ (before-point), or $\mathbf{t}_{p_\mathrm{a}} = [0\ 1]$ (after-point).

(SV+AP) When combining both kinds of visibility information, before-points p_b and after-points p_a are given the same sightline vector as their reference point, *i.e.*, $\mathbf{v}_{p_\mathrm{b}} = \mathbf{v}_{p_\mathrm{a}} = \mathbf{v}_p$, and we take as input $I_p = (X_p \oplus \mathbf{v}_p \oplus \mathbf{t}_p) \in \mathbb{R}^8$.

While holding a similar kind of information, no augmentation can be reduced to the other one. SVs alone are not enough to place APs, and APs alone, as they are not associated to their observed point in P, cannot determine SVs (cf. Figure 2).

D. Modifying an Existing Architecture

We can adapt most DSR networks to handle visibility-augmented point clouds with only few modifications:

- We change the input size (number of channels) of the first layer of the network (generally an encoder), increasing it by 2, 3 or 5, depending on the augmentation.
- We directly add auxiliary points to the point cloud, thus tripling the number of input points. For methods based on neighboring point sampling, we add auxiliary points after sampling for more efficiency.

The batch size may need to be adjusted to fit a larger point cloud in memory, but the rest of the network stays unchanged. Its size is mostly unaltered (*e.g.*, +0.005% for ConvONet [25]).

IV. Experiments

To assess our proposal, we first detail our simple adaptation of six different DSR baseline networks to leverage our visibility information, then compare the quality of the reconstructed surfaces and analyze the generalization ability of the networks.

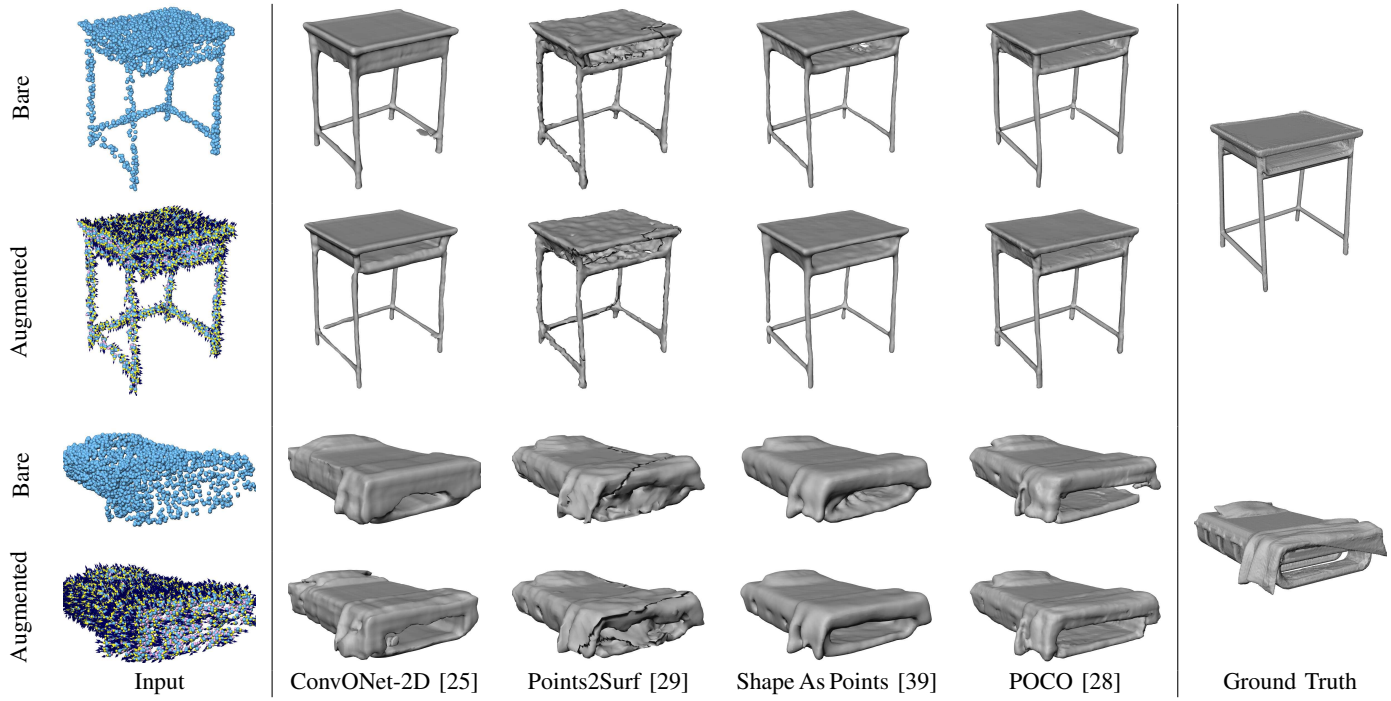


Fig. 3. **Object-Level Reconstruction.** Reconstructed shapes from the ModelNet10 test set using four different DSR methods trained on ModelNet10. Top rows of each object use the bare point cloud as input, and bottom rows use the point cloud augmented with visibility information.

A. DSR Baselines

1) ConvONet [25]: This method first extracts point features and projects them on three 2D grids, or one 3D grid (variant). 2D or 3D grid convolutions then create features capturing local occupancy. Last, the occupancy of a query-point is estimated after interpolating grid features. We consider the 3×64^2 2D-plane encoder and the 64^3 3D-volume variant. To adapt them, we change the input size of the point encoder's first layer.

2) Points2Surf [29]: This method predicts both the occupancy of a query point and its unsigned distance to the surface. It uses both a local query-point neighborhood and a global point-cloud sampling. We use the best-performing variant (uniform global sampling, no spatial transformer). To adapt it, we increase the input size of the first layer of both the local and global encoders, and when a point is sampled, locally or globally, we add its two auxiliary points on the fly.

3) Shape As Points [39]: For each input point, the method estimates its normal as well as k point offsets that are used to correct and densify the point cloud. The resulting point cloud of size $k|P|$ is then fed to a differentiable Poisson solver [31]. To adapt the method, we change the input size of the first layer of the encoder, and of the normal and offset decoders as they also input the point cloud. We directly add auxiliary points as input, whose normal and offsets will thus be computed too.

4) Local Implicit Grids (LIG) [5]: This method trains an auto-encoder from dense point cloud patches. For inference, a given sparse patch with oriented normals is first augmented, close to our idea, with 10 new points along each normal; then reconstruction uses latent vectors minimizing a decoder-based training loss, and a post-processing removes falsely-enclosed volumes. As training code is unavailable, we use the model pretrained on ShapeNet (without noise). For oriented normals, we use Jets [40] oriented with a minimum spanning tree [7], as in [23]. To exploit visibility, we replace normals with sightline vectors; we do not add (more) auxiliary points.

5) POCO [28]: This method extracts point features using point cloud convolution [41], then estimates the occupancy of a query point with a learning-based interpolation on nearest neighbors. To adapt it, we increase the input size of the first layer and add auxiliary points on the fly only in the first layer.

6) DGNN [6]: This method uses a graph neural network to estimate the occupancy of Delaunay cells in a point cloud tetrahedralization. A graph-cut-based optimization then reinforces global consistency. The method, which already uses visibility, outperforms other traditional reconstruction methods that use visibility information. As it already exploits visibility, we do not alter it, but use it as baseline for comparison.

For all methods, unless otherwise stated, training and evaluation are unchanged; we keep the value of the hyperparameters used in the original papers. When Marching cubes [42] are needed for surface extraction, we use a grid resolution of 128^3.

B. Datasets

We consider a variety of object and scene datasets, both synthetic and real, to show the versatility of our approach.

1) ModelNet10: We use the official train/test splits of all 10 object classes of ModelNet10 [43]. We hold out 10% of the train set for validation. We synthetically scan the models by placing 10 random range scanners in two bounding spheres around the objects and shooting random rays to a sphere inscribed within the convex hull of the object. We sample 3 000

TABLE I
ABLATION STUDY.

The vanilla model of ConvONet trained and tested on ModelNet10 with different ways to add visibility or normal information.

Model	SV	AP	IoU ↑
ConvONet-2D (3×64^2) [25]			0.853
+ sightline vectors (SV) only	✓		0.871
+ auxiliairy points (AP) only		✓	0.881
+ both SV and AP	✓	✓	**0.886**
+ sensor position	S_p		0.870
+ unnormalized SV	$S_p - X_p$		0.870
+ estim. normals / estim. orientation	Jets [40] / MST [7]		0.853
+ estim. normals / sensor orientation	Jets [40] / sensor-based [7]		0.868
+ true normals	GT normals		0.879

points per object and add Gaussian noise with zero mean and standard deviation 0.005 as in [25].

2) ShapeNet: We study the generalizability of models trained on ModelNet10 by testing on 100 shapes per class from the ShapeNet [44] test set of Choy *et al.* [45] (9 out of 13 classes are not represented in ModelNet10). We use the same scanning procedure as for ModelNet10.

3) Synthetic Room: We use the train/val/test splits of Synthetic Rooms [25]. For virtual scanning, we only place sensors in the upper hemispheres, and scan $10\,000$ points as in [25].

4) SceneNet: We test on a few synthetic scenes of SceneNet [46] using the given virtual scans, voxel-decimated to 1 cm.

5) ScanNet: We test on a few real scenes of ScanNet [47] using the provided real RGB-D scans, voxel-decimated to 2 cm.

6) Tanks and Temples: We use the real LiDAR point cloud of the *Ignatius* statue from the Tanks and Temples dataset [48].

7) Middlebury: We use an MVS point cloud of the *TempleRing* from Middlebury [49], made with OpenMVS [50].

C. Metrics

We report volumetric intersection over union (IoU), mean Chamfer distance $\times 100$ (CD) and normal consistency (NC).

D. Ablation Study

To validate our design, we compare in Table I various ways to add visibility information to the vanilla model of ConvONet.

Independently, SVs and APs significantly improve performance (+1.8 and +2.8 IoU pts). A reason why APs are more profitable could be that the network is tailored for points, not points with sightline features. While SVs and APs capture a similar kind of information, they are, however, complementary: combining them is even more beneficial (+3.5 IoU pts). Our general interpretation is that SVs help to decide whether a locally "thin" point cloud is to be considered as a noisy scan of a single surface, or as a (less noisy) scan on both sides of a thin surface. They thus have an impact on local shape topology, which can bring a notable gain. Auxiliary points convey similar information, but also contribute more directly to refine the surface position. Replacing SVs by the sensor position or by the unnormalized point-sensor vector gives essentially the same performance than our unit vector.

TABLE II
OBJECT-LEVEL RECONSTRUCTION.

DSR methods trained and tested on ModelNet10, with and without sightline vectors (SV) or auxiliary points (AP). † Trained on ShapeNet.

Model	SV	AP	IoU ↑	CD ↓	NC ↑
ConvONet-2D [25]			0.853	0.618	0.934
ConvONet-2D [25]	✓		0.871	0.557	0.936
ConvONet-2D [25]	✓	✓	**0.886**	**0.518**	**0.943**
ConvONet-3D [25]			0.885	0.493	0.949
ConvONet-3D [25]	✓		0.911	0.424	0.956
ConvONet-3D [25]	✓	✓	**0.913**	**0.423**	**0.957**
Points2Surf [29]			0.842	0.590	0.890
Points2Surf [29]	✓		**0.859**	**0.544**	0.896
Points2Surf [29]	✓	✓	0.856	0.548	**0.897**
Shape As Points [39]			0.903	0.438	0.948
Shape As Points [39]	✓		0.907	0.430	0.950
Shape As Points [39]	✓	✓	**0.913**	**0.414**	**0.953**
POCO [28]			0.907	0.422	0.945
POCO [28]	✓		0.915	0.408	**0.950**
POCO [28]	✓	✓	**0.917**	**0.406**	**0.950**
† LIG [5]			–	0.974	0.849
† LIG [5]	✓		–	**0.880**	**0.882**
DGNN [6]	✓		0.866	0.543	0.884

This can be explained by the fact that our scanning procedure does not introduce significant variation in terms of distance to the sensor. Yet, for real world acquistions, with a larger range of sensor distances, normalizing the SV ensures more stability.

Adding SVs outperforms estimated normals [40] with estimated orientation [7], and even estimated normals with sensor-based orientation. While using ground-truth normals is slightly more beneficial than SVs, combining SV+AP yields the best overall performance, which highlights the richness of our visibility information.

We also experiment with adding more than two auxiliary points: (i) at distance $0.5d$ or $2d$, (ii) at the midpoint between sensor and point, or (iii) as grazing points, estimated by densely sampling the sightlines with auxiliary points and keeping the ones close to an input point. None of these strategies brought significant improvements over adding two points at distance d on both sides of the real point.

E. Object-Level Reconstruction

Table II reports the performance on ModelNet10 of various models, with and without sightline vectors or auxiliary points.

ConvONet (both planar and volumetric) gains about +3 IoU pts with visibility information. The resulting surface is more accurate, especially in concave parts, as illustrated in Figure 3.

Points2Surf improves with sightline vectors, but auxiliary points do not improve further: the sensor vectors are enough to resolve ambiguities for the occupancy estimation, but distance estimation does not further benefit from auxiliary points.

Shape As Points benefits from sightline vectors, although not as much as other methods, probably because the model also estimates normals which provide a similar information as visibility. Still, adding auxiliary points further gains +0.6 IoU pts, yielding more complete and smoother surfaces.

TABLE III
SCENE-LEVEL RECONSTRUCTION.
ConvONet trained and tested in sliding-window mode on Synthetic Rooms.

Model	SV	AP	IoU ↑	CD ↓	NC ↑
ConvONet-3D [25]			0.805	0.598	0.906
ConvONet-3D [25]	✓	✓	**0.832**	**0.569**	**0.911**

TABLE IV
OUT-OF-DOMAIN OBJECT-LEVEL RECONSTRUCTION.
DSR methods trained on ModelNet10 and tested on ShapeNet, with and without sightline vectors (SV) or auxiliary points (AP). † Trained on ShapeNet.

Model	SV	AP	IoU ↑	CD ↓	NC ↑
† ConvONet-2D [25]			0.852	0.560	0.929
ConvONet-2D [25]			0.685	0.979	0.878
ConvONet-2D [25]	✓		0.667	1.042	0.833
ConvONet-2D [25]	✓	✓	**0.750**	**0.891**	**0.878**
ConvONet-3D [25]			0.628	0.972	0.885
ConvONet-3D [25]	✓		0.759	0.724	0.905
ConvONet-3D [25]	✓	✓	**0.854**	**0.554**	**0.925**
Points2Surf [29]			0.807	0.561	0.876
Points2Surf [29]	✓		**0.836**	**0.516**	0.886
Points2Surf [29]	✓	✓	0.833	0.522	**0.887**
† Shape As Points [39]			0.838	0.577	0.923
Shape As Points [39]			0.494	0.997	0.859
Shape As Points [39]	✓		0.749	0.843	0.881
Shape As Points [39]	✓	✓	**0.821**	**0.617**	**0.919**
POCO [28]			0.391	1.119	0.839
POCO [28]	✓		**0.832**	**0.618**	**0.901**
POCO [28]	✓	✓	0.815	0.635	0.887
DGNN [6]	✓		0.844	0.549	0.854

POCO similarly benefits +1 IoU pts from sightline vectors but not much from the further addition of auxiliary points. While sightline vectors help for surface orientation, POCO is already accurate enough for APs to bring little refinement.

LIG produces poor results, likely because the only available model is trained on ShapeNet, with uniform sampling, little or no noise, and because oriented normals are only estimated. We cannot report IoU because LIG's post-processing creates holes in some objects. Yet, replacing the estimated normals by sightline vectors improves the predicted surface.

DGNN, which already exploits visibility and outperforms ConvONet-2D and Points2Surf, is outdistanced on this dataset by methods that use our augmented point clouds.

Running time is mostly unaffected when adding sightline vectors. The effect of auxiliary points depends on the method. For ConvONet and POCO, it is negligible ($<+2\%$); running time is $\times$ 2.25 for Points2Surf and $\times$ 1.25 for Shape As Points.

F. Scene-Level Reconstruction

To study the impact of visibility information at scene level, we train and test ConvONet on Synthetic Rooms, in sliding-window mode [25]. We report quantitative results in Table III and qualitative results in the supplementary material. The model gains almost +3 IoU pts with visibility information, showing that benefits scale to scenes, not just objects.

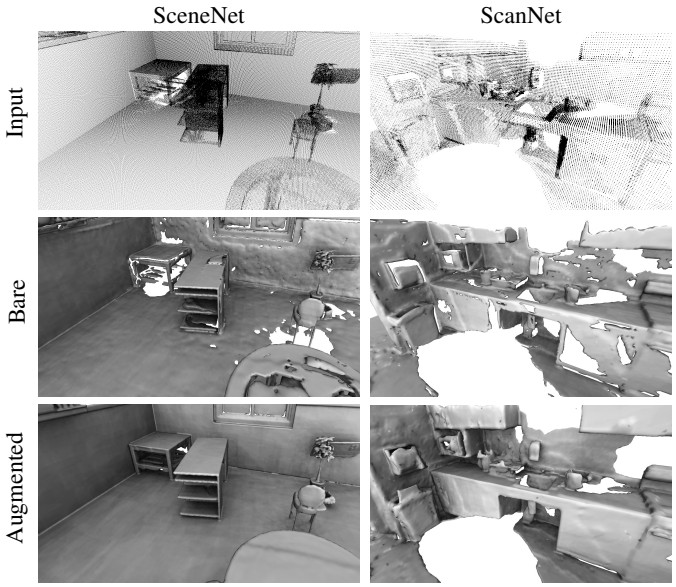


Fig. 4. **Out-of-Domain Scene-Level Reconstruction.** POCO trained on ModelNet10, with and without visibility information, is run on scenes from SceneNet (synthetic RGB-D scan) and ScanNet (real RGB-D scan).

G. Generalization to New Domains

To evaluate the impact of added visibility on the generalization capability of DSR methods, we train on ModelNet10 and test on ShapeNet (Table IV).

We observe that ConvONet, Shape As Points and POCO trained with visibility information generalize much better on the new objects and classes, with a gain up to +44 IoU pts. For comparison, we also show the scores of official models trained on ShapeNet, although trained on uniformly sampled points rather than virtual scans, which explains the drop of performance compared to the numbers in the papers [25], [39]. Points2Surf also improves by up to +3 IoU pts with added sightline vectors, but not further with APs.

The increased generalization capability of the models is also validated when reconstructing surfaces from real-world scans obtained with LiDAR or MVS. In Figures 4 and 5, we show that networks using visibility information can reconstruct a more accurate and more complete surface. The reason for the largely improved volumetric IoU when using visibility information is illustrated in Figure 6. For out-of-domain reconstructions, the baseline methods often predict hollow shapes, i.e., empty space enclosed inside an object. This leads to backfaces behind the real surface and a poor volumetric IoU. On the contrary, our models, trained on visibility-augmented point clouds, learn to distinguish between empty and full space more reliably and do not produce such artifacts.

V. LIMITATIONS AND PERSPECTIVES

The position of auxiliary points depends on parameter d, which is the average distance, across the whole scene, from a point to its nearest neighbor. To better handle point density variations, it could be set locally rather than globally. Besides,

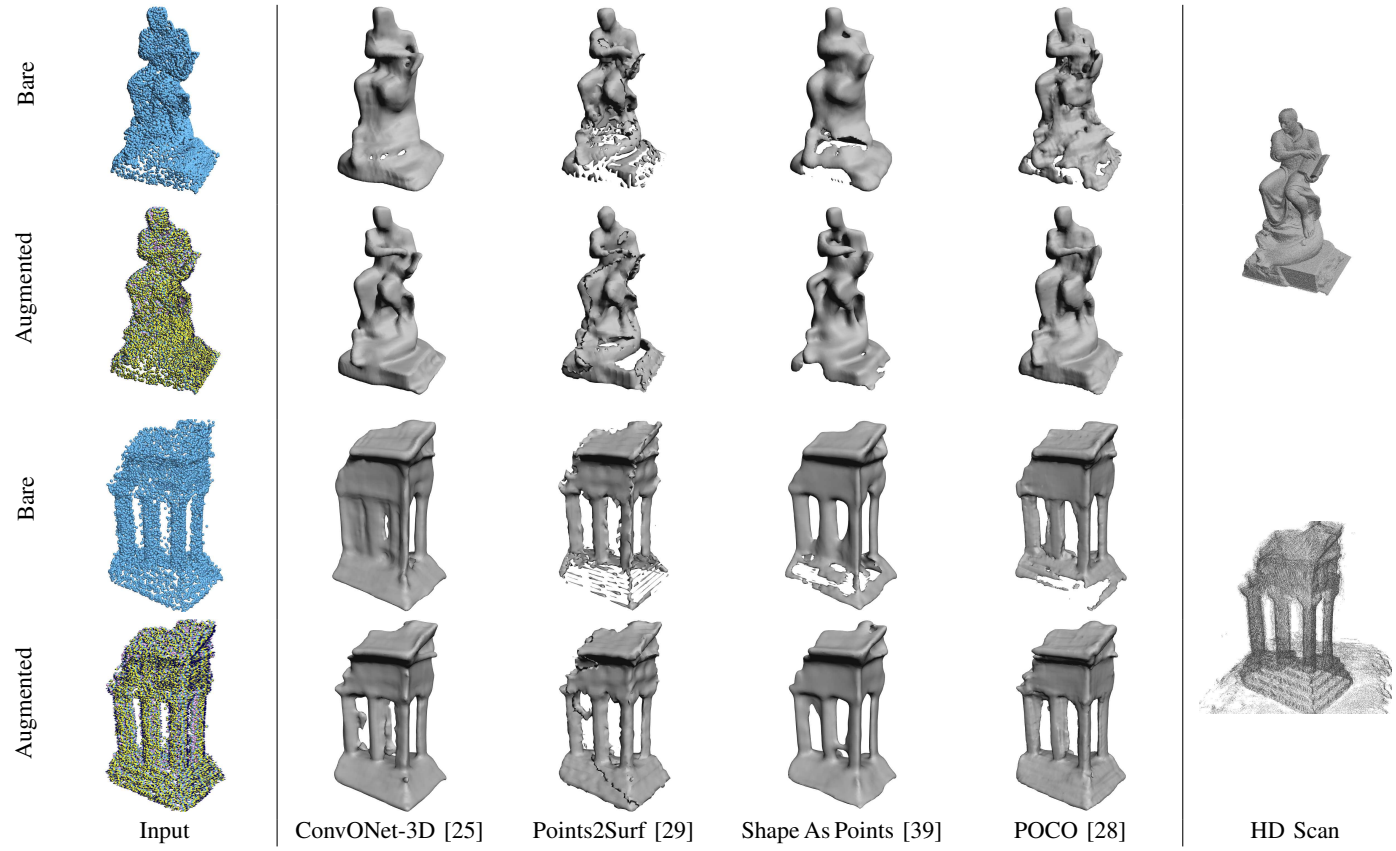


Fig. 5. **Out-of-Domain Object-Level Reconstruction.** Reconstructed shapes from a LiDAR point cloud (top, *Ignatius* from Tanks And Temples) and a MVS point cloud (bottom, *TempleRing* from Middlebury) using four different DSR methods trained on ModelNet10. Top rows of each object use the bare point cloud as input, and bottom rows use the point cloud augmented with visibility information. *HD Scan* is a high-density point cloud.

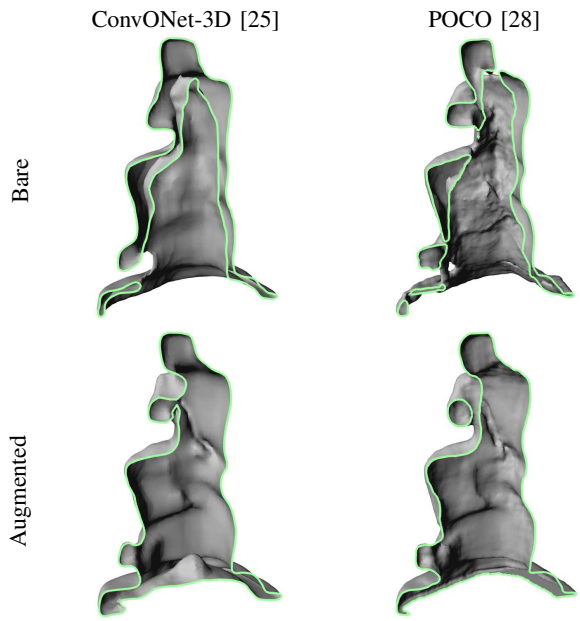


Fig. 6. **Cut of Out-of-Domain Object Reconstruction.** A cut (along the green curve) of the reconstructed surface of *Ignatius*. The reconstructions from the bare point cloud (top row) include empty space enclosed inside the object with backfaces, leading to a poor volumetric IoU. The reconstructions from the point cloud augmented with visibility information (bottom row) include only one surface, close to the input points.

as this positioning is also sensitive to sampling noise, d could also be directly adjusted after noise estimation.

Our current approach only associates each point with a single sensor, while MVS points typically have several. A more efficient and versatile approach than simply duplicating sightlines is still an open issue.

Last, we resort to virtual scans because current 3D reconstruction benchmarks do not provide sensor positions. While we show that using our augmented point clouds allows common architectures to successfully generalize from virtual to real scenes, our training set may fail to replicate some challenging configurations encountered using actual sensors.

VI. Conclusion

The sensor poses are often ignored in point cloud processing, even though available with most acquisition technologies. We present two straightforward ways to exploit sensor positions to augment point clouds with visibility information. Our experiments show that various deep surface reconstruction methods can be adapted with minimal effort to exploit these visibility-augmented point clouds, resulting in improved accuracy and completeness of reconstructed surfaces, as well as a substantial increase in generalization capability.

Acknowledgments: This work was partially funded by the ANR-17-CE23-0003 BIOM grant.

REFERENCES

[1] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[3] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or, "Point2Mesh: A self-prior for deformable meshes," *ACM Transaction on Graphics*, 2020.

[4] A. Dai, C. Diller, and M. Nießner, "SG-NN: Sparse generative neural networks for self-supervised scene completion of RGB-D scans," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[5] C. M. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser, "Local implicit grid representations for 3D scenes," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] R. Sulzer, L. Landrieu, R. Marlet, and B. Vallet, "Scalable surface reconstruction with delaunay-graph neural networks," *Eurographics Symposium on Geometry Processing (SGP)*, 2021.

[7] N. Schertler, B. Savchynskyy, and S. Gumhold, "Towards globally optimal normal orientations for large point clouds," *Computer Graphics Forum (CFG)*, 2017.

[8] P. Labatut, J. P. Pons, and R. Keriven, "Robust and efficient surface reconstruction from range data," *Computer Graphics Forum (CGF)*, 2009.

[9] H. H. Vu, P. Labatut, J. P. Pons, and R. Keriven, "High accuracy and visibility-consistent dense multiview stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2012.

[10] M. Jancosek and T. Pajdla, "Multi-view reconstruction preserving weakly-supported surfaces," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[11] L. Caraffa, M. Brédif, and B. Vallet, "3D watertight mesh generation with uncertainties from ubiquitous data," in *Asian Conference on Computer Vision (ACCV)*, 2017.

[12] A. Bódis-Szomorú, H. Riemenschneider, and L. Van Gool, "Efficient volumetric fusion of airborne and street-side data for urban reconstruction," in *International Conference on Pattern Recognition (ICPR)*, 2016.

[13] M. Jancosek and T. Pajdla, "Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces," *International Scholarly Research Notices*, 2014.

[14] Y. Zhou, S. Shen, and Z. Hu, "Detail preserved surface reconstruction from point cloud," *Sensors*, 2019.

[15] L. Caraffa, Y. Marchand, M. Brédif, and B. Vallet, "Efficiently distributed watertight surface reconstruction," in *3DV*, 2021.

[16] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, "Deep local shapes: Learning local SDF priors for detailed 3D reconstruction," in *European Conference on Computer Vision (ECCV)*, 2020.

[18] M. Michalkiewicz, J. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson, "Implicit surface representations as layers in neural networks," in *International Conference on Computer Vision (ICCV)*, 2019.

[19] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *International Conference on Machine Learning (ICML)*, 2020.

[20] M. Atzmon and Y. Lipman, "SAL: Sign agnostic learning of shapes from raw data," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[21] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit functions in feature space for 3D shape reconstruction and completion," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[22] M. Atzmon and Y. Lipman, "SALD: sign agnostic learning with derivatives," in *International Conference on Learning Representations (ICLR)*, 2021.

[23] W. Zhao, J. Lei, Y. Wen, J. Zhang, and K. Jia, "Sign-agnostic implicit learning of surface self-similarities for shape modeling and reconstruction from raw point clouds," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[24] J. Chibane, A. Mir, and G. Pons-Moll, "Neural unsigned distance fields for implicit function learning," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[25] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *European Conference on Computer Vision (ECCV)*, 2020.

[26] J. Tang, J. Lei, D. Xu, F. Ma, K. Jia, and L. Zhang, "SA-ConvONet: Sign-agnostic optimization of convolutional occupancy networks," in *International Conference on Computer Vision (ICCV)*, 2021.

[27] B. Ummenhofer and V. Koltun, "Adaptive surface reconstruction with multiscale convolutional kernels," in *International Conference on Computer Vision (ICCV)*, 2021.

[28] A. Boulch and R. Marlet, "POCO: Point convolution for surface reconstruction," *arXiv preprint arXiv:2201.01831*, 2022.

[29] P. Erler, S. Ohrhallinger, N. Mitra, and M. Wimmer, "Points2Surf: Learning implicit surfaces from point clouds," in *European Conference on Computer Vision (ECCV)*, 2020.

[30] N. Sharp and M. Ovsjanikov, ""pointtrinet: Learned triangulation of 3d point sets"," in *European Conference on Computer Vision (ECCV)*, 2020.

[31] M. Kazhdan and H. Hoppe, "Screened Poisson surface reconstruction," *ACM Transaction on Graphics.*, 2013.

[32] F. Williams, M. Trager, J. Bruna, and D. Zorin, "Neural splines: Fitting 3d surfaces with infinitely-wide neural networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[33] S. König and S. Gumhold, "Consistent propagation of normal orientations in point clouds." in *International Workshop on Vision, Modeling, and Visualization (VMV)*, 2009.

[34] G. Metzer, R. Hanocka, D. Zorin, R. Giryes, D. Panozzo, and D. Cohen-Or, "Orienting point clouds with dipole propagation," *ACM Transactions on Graphics (TOG)*, 2021.

[35] D. Paschalidou, O. Ulusoy, C. Schmitt, L. Van Gool, and A. Geiger, "Raynet: Learning volumetric 3d reconstruction with ray potentials," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[36] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision (ECCV)*, 2020.

[37] M. Oechsle, S. Peng, and A. Geiger, "Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction," in *International Conference on Computer Vision (ICCV)*, 2021.

[38] S. Song, Z. Cui, and R. Qin, "Vis2mesh: Efficient mesh reconstruction from unstructured point clouds of large scenes with learned virtual view visibility," in *International Conference on Computer Vision (ICCV)*, 2021.

[39] S. Peng, C. M. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, "Shape as points: A differentiable poisson solver," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[40] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Computer Aided Geometric Design*, 2005.

[41] A. Boulch, "Convpoint: Continuous convolutions for point cloud processing," *Computers & Graphics*, 2020.

[42] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, 1987.

[43] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[44] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: an information-rich 3D model repository," Stanford University, Princeton University, Toyota Technological Institute at Chicago, Tech. Rep., 2015.

[45] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," in *European Conference on Computer Vision (ECCV)*, 2016.

[46] A. Handa, V. Patraucean, S. Stent, and R. Cipolla, "SceneNet: An annotated model generator for indoor scene understanding," in *International Conference on Robotics and Automation (ICRA)*, 2016.

[47] A. Dai, A. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[48] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and Temples," *ACM Transactions on Graphics (TOG)*, 2017.

[49] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[50] D. Cernea, "OpenMVS: Multi-view stereo reconstruction library," 2020. [Online]. Available: https://cdcseacave.github.io/openMVS

[51] J. Huang, Y. Zhou, and L. Guibas, "Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups," *arXiv preprint arXiv:2005.11621*, 2020.

[52] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[53] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, "Large scale multi-view stereopsis evaluation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

SUPPLEMENTARY MATERIAL

In this supplementary document, we first provide additional information about the datasets that we use (Section VII), formal definitions of the evaluation metrics (Section VIII) and additional quantitative and qualitative results (Section IX). Our code, data and pretrained models can be found online: https://github.com/raphaelsulzer/dsrv-data .

VII. DATASETS

A. Scanning procedure

In Figure 7, we represent a visualization of our scanning procedure. Viewpoints and ray target points are uniformly sampled on the surface of the spheres.

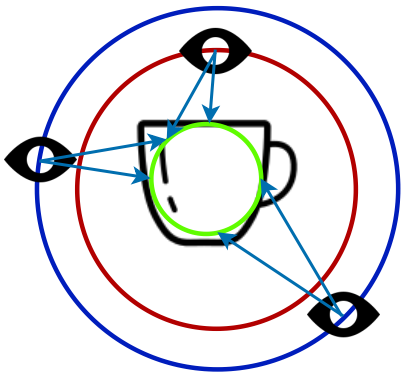


Fig. 7. **Scanning Procedure.** We randomly place sensors on two spheres (red and blue) around the object, and consider rays aiming at uniformly sampled points on a sphere inscribed in the convex hull of the object (green).

B. ModelNet

We use the official ModelNet10 dataset and make the models watertight using ManifoldPlus [51]. We scan the watertight models using the procedure described above.

C. ShapeNet

We use the watertight models provided[1] by the authors of Occupancy Networks [2] and scan the models using the procedure described above. We apply a transformation to the models (and scans) to match their orientation to the orientation of the ModelNet10 objects (except for networks marked with † in Table IV, which were trained with the original orientation).

D. Synthetic Rooms Dataset

We use the watertight scenes provided[2] by the authors of ConvONet [25]. We scan the scenes using the procedure described above, limiting the sensors to lie in the upper hemisphere only.

E. Tanks And Temples, Middlebury and DTU

For the reconstructions in Figure 5, 6 and Figure 8, we downsample the input point clouds to 10,000 points.

[1]https://s3.eu-central-1.amazonaws.com/avg-projects/occupancy_networks/data/watertight.zip

[2]https://s3.eu-central-1.amazonaws.com/avg-projects/convolutional_occupancy_networks/data/room_watertight_mesh.zip

VIII. METRICS

We evaluate the quality of reconstructions with the volumetric IoU (IoU), symmetric Chamfer distance (CD) and normal consistency (NC).

Let $\mathcal{M}_\mathcal{G}$ be the ground truth mesh and $\mathcal{M}_\mathcal{P}$ be the reconstructed mesh. The volumetric IoU is defined as:

$$\text{IoU}\left(\mathcal{M}_\mathcal{G}, \mathcal{M}_\mathcal{P}\right) = \frac{\text{volume}(\mathcal{M}_\mathcal{G} \cap \mathcal{M}_\mathcal{P})}{\text{volume}(\mathcal{M}_\mathcal{G} \cup \mathcal{M}_\mathcal{P})},$$

We approximate volumetric IoU by sampling $100,000$ points in the union of the bounding boxes of $\mathcal{M}_\mathcal{G}$ and $\mathcal{M}_\mathcal{P}$.

To compute the Chamfer distance and normal consistency, we sample a set of points S_G on the ground-truth mesh and a set of points S_P on the reconstructed mesh with $|S_G| = |S_P| = 100,000$. We approximate the two-sided Chamfer distance between $\mathcal{M}_\mathcal{G}$ and $\mathcal{M}_\mathcal{P}$ as follows:

$$\begin{aligned}\text{CD}(\mathcal{M}_\mathcal{G}, \mathcal{M}_\mathcal{P}) = &\frac{1}{2|S_G|} \sum_{x \in S_G} \min_{y \in S_P} ||x - y||_2 \\ &+ \frac{1}{2|S_P|} \sum_{y \in S_P} \min_{x \in S_G} ||y - x||_2\end{aligned}$$

Let $n(x)$ be the unit normal associated to a point x taken on a mesh, and $\langle \cdot, \cdot \rangle$ the Euclidean scalar product in $\mathbb{R}^3$. Normal consistency is defined as:

$$\begin{aligned}\text{NC}(\mathcal{M}_\mathcal{G}, \mathcal{M}_\mathcal{P}) = &\frac{1}{2|S_G|} \sum_{x \in S_G} \left\langle n(x), n\left(\operatorname*{argmin}_{y \in S_P} ||x - y||_2\right)\right\rangle \\ &+ \frac{1}{2|S_P|} \sum_{y \in S_P} \left\langle n(y), n\left(\operatorname*{argmin}_{x \in S_G} ||y - x||_2\right)\right\rangle\end{aligned}$$

IX. ADDITIONAL RESULTS

A. Runtimes

In Table V, we report detailed runtimes for the tested methods, with and without visibility information.

Adding sightline vectors does not significantly increase the runtime for any of the tested methods. The effect of auxiliary points depends on the method. For ConvONet, most of the processing time is spent computing grid features. The encoding of 3d points is performed by a small PointNet network [52], whose runtime is only a small fraction of the total time. As a consequence, adding APs does not incur significant changes in computation time. In contrast, Points2Surf uses a large point encoding network and is 2.2 times slower with APs. Shape As Points is 1.3 times slower due to the fact that we decode 3 times as many points as the baseline method. POCO is also essentially unaffected by the addition of auxiliary points as they only impact the first (small) layer of the point-convolution backbone.

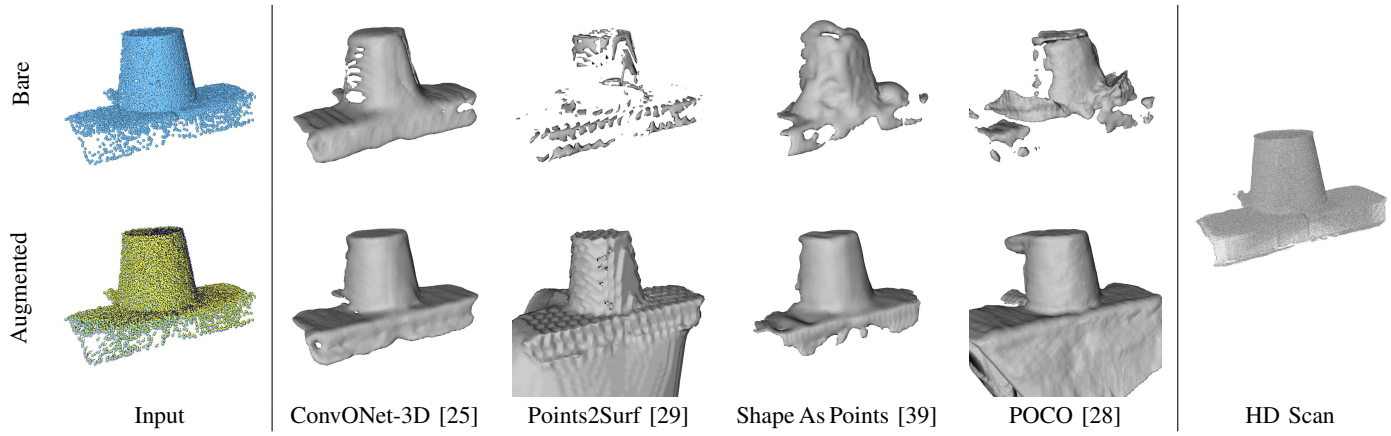


Fig. 8. **Out-of-Domain Object-Level Reconstruction.** Reconstructed shape from a MVS point cloud of *scan1* from DTU, using four different DSR methods trained on ModelNet10. The top row uses the bare point cloud as input, and the bottom row uses the point cloud augmented with visibility information. *HD Scan* is a high-density point cloud.

TABLE V
RUNTIMES FOR OBJECT-LEVEL RECONSTRUCTION.

Average times (in seconds) for reconstructing one object from a point cloud of 3 000 points with and without sightline vectors (SV) or auxiliary points (AP). MC is marching cubes. Times are averaged over the ModelNet10 test set.

Model	SV	AP	Encoding	Decoding	MC	Total
ConvONet-2D [25]			0.016	0.25	0.17	0.44
ConvONet-2D [25]	✓		0.016	0.27	0.17	0.47
ConvONet-2D [25]	✓	✓	0.016	0.26	0.17	0.45
Points2Surf [29]			69.06		11.51	80.57
Points2Surf [29]	✓		71.92		11.35	83.27
Points2Surf [29]	✓	✓	173.2		11.41	184.7
Shape As Points [39]			0.022	0.017	0.047	0.088
Shape As Points [39]	✓		0.023	0.017	0.046	0.086
Shape As Points [39]	✓	✓	0.024	0.041	0.047	0.114
POCO [28]			0.088	13.72	0.33	15.74
POCO [28]	✓		0.091	13.68	0.33	15.66
POCO [28]	✓	✓	0.093	13.70	0.33	15.67

B. DTU Dataset

In Figure 8, we show the reconstruction of an MVS point cloud, generated with OpenMVS, of *scan1* from the DTU dataset [53]. The point cloud represents an open scene, while all methods were trained on the closed ModelNet10 objects. The methods using our augmented point clouds with visibility information cope much better with this domain shift.

C. ShapeNet

We show the results of object-level reconstruction on ShapeNet in Figure 9 from methods trained on ModelNet10. All methods benefit from added visibility information. In particular, ConvONet produces very accurate and complete surfaces of the unseen shape classes.

A common problem for most baseline methods is the false reconstruction of hollow shapes with enclosed outside space. Using visibility information can address this issue and the artifacts then do not occur.

D. ModelNet10

In Figure 10, we represent additional results of object-level reconstruction on ModelNet10. Concave parts of the objects are frequently reconstructed more accurately by methods using point clouds with visibility information. Surfaces also tend to be more complete when visibility information is used.

E. Synthetic Rooms Dataset

In Figure 11, we show the reconstruction results of ConvONet on Synthetic Rooms, with and without visibility information. In contrast to [25], we evaluate in sliding window mode, which explains the difference with figures in [25]. Improvements here are visually not as obvious as with other datasets.

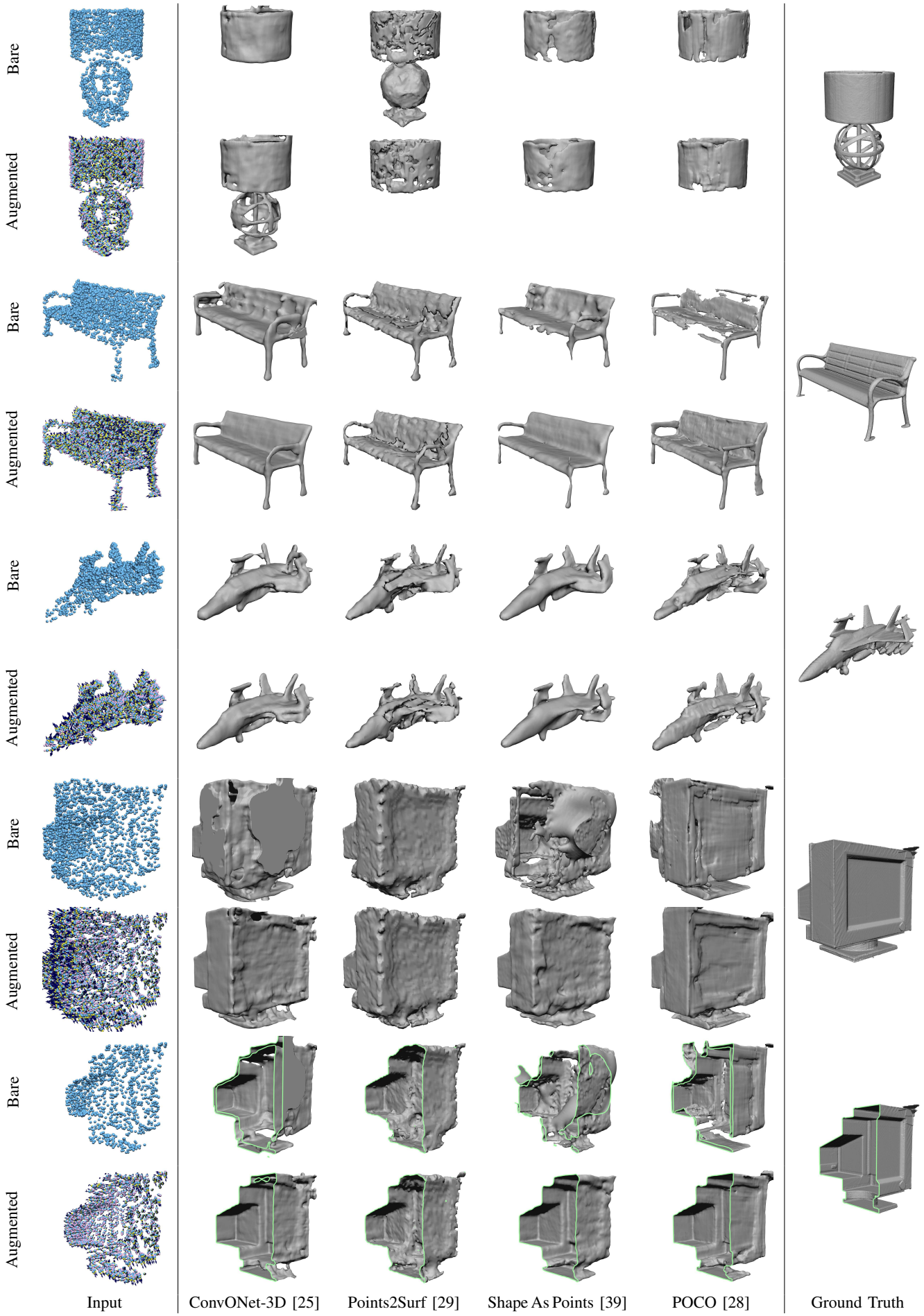


Fig. 9. **Object-Level Reconstruction on ShapeNet.** Reconstructed shapes from the ShapeNet test set using four different DSR methods trained on ModelNet10. Top rows of each object use the bare point cloud as input, and bottom rows use the point cloud augmented with visibility information. The last two rows show a cut of the reconstructions that are shown on the two other rows immediately above.

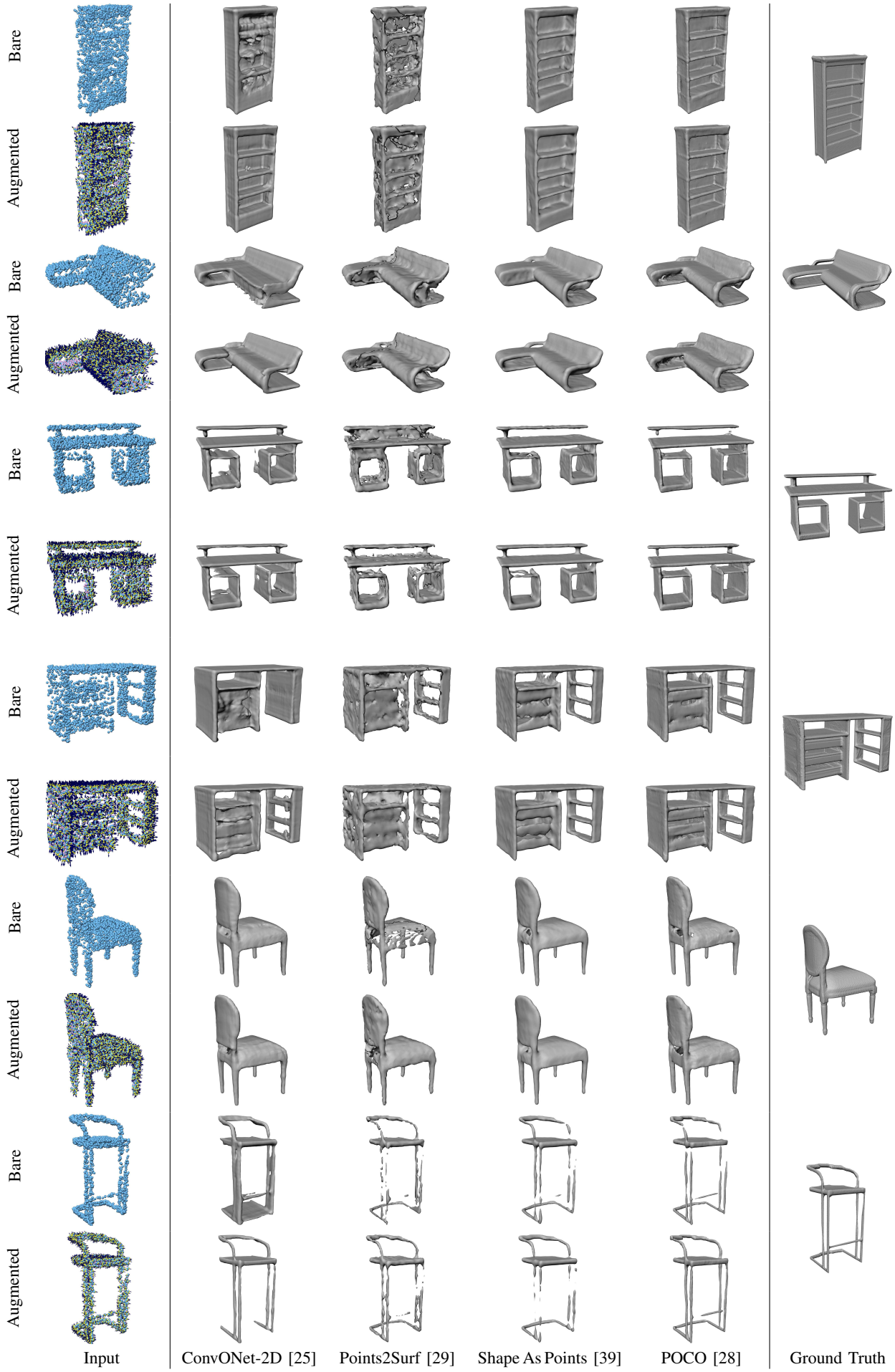


Fig. 10. **Object-Level Reconstruction On ModelNet10.** Reconstructed shapes from the ModelNet10 test set using four different DSR methods trained on ModelNet10. Top rows of each object use the bare point cloud as input, and bottom rows use the point cloud augmented with visibility information.

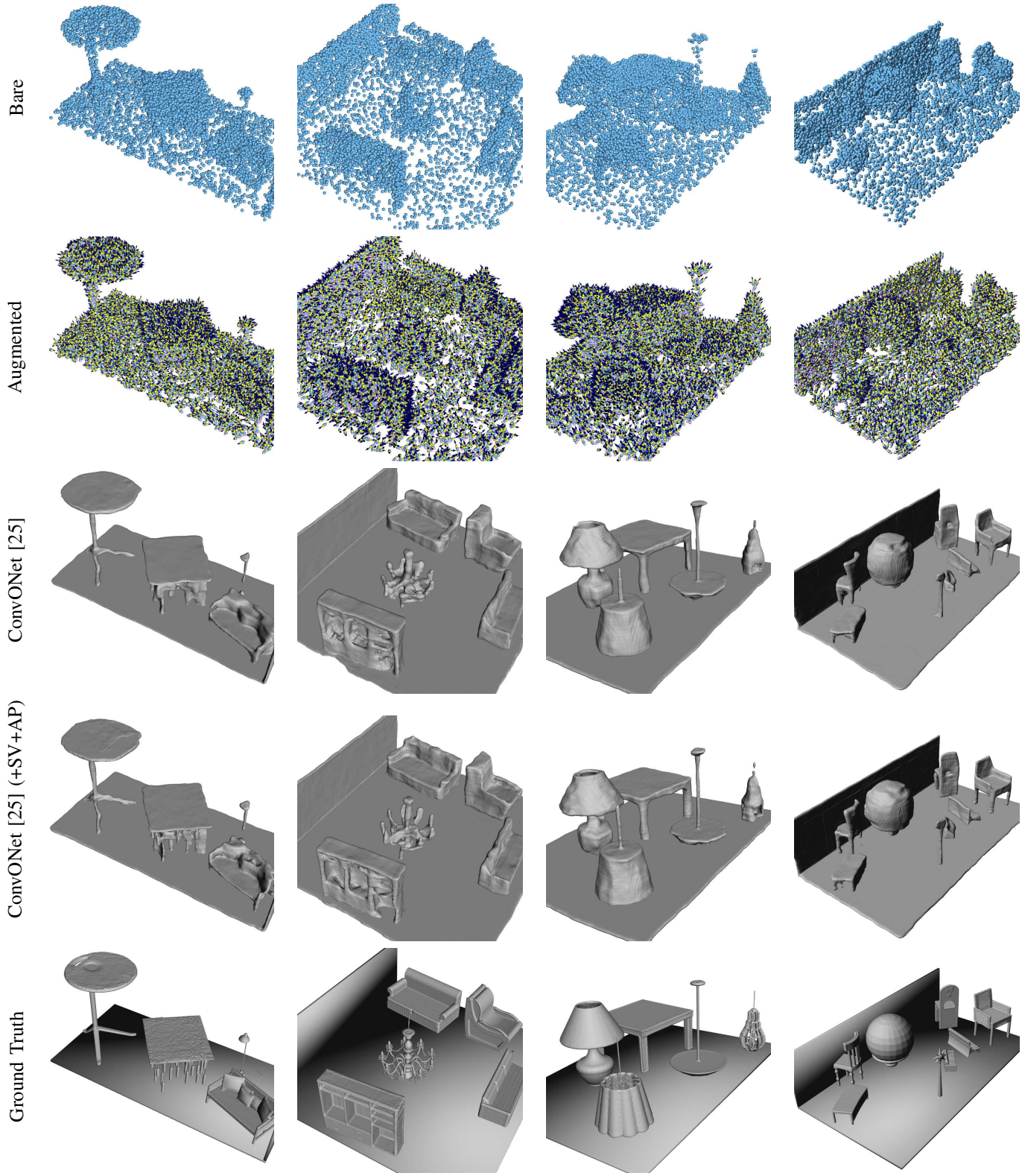


Fig. 11. **Scene-Level Reconstruction on Synthetic Rooms.** Reconstructed scenes of the synthetic room dataset using ConvONet [25] in sliding-window mode, with and without visibility information.