# POCO: Point Convolution for Surface Reconstruction — Supplementary material —

Alexandre Boulch[1] Renaud Marlet[1,2]

[1]Valeo.ai, Paris, France [2]LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-Vallée, France



## A. Implementation details

**Code avalability.** The code of our method is available at https://github.com/valeoai/POCO.

**Framework and hardware.** Our code uses PyTorch as deep learning framework. All experiments were done with a single NVIDIA RTX 2080 Ti GPU with 11GB memory.

**Backbone.** We used FKAConv [4] as convolutional backbone, with default parameters (number of layers, number of layer channels). Only the latent vector size $n$, i.e., the output dimension of the backbone, was changed. It was set to 32, which is also the output dimension of all linear layers of the occupancy decoder (except the last one, which outputs the occupancy). As a comparison, methods like ConvONet [33] and LIG [24] also use latent vectors of size 32.

**Architecture.** The network architecture is described in Figure 5 of the main paper. We phrase here some parts of it.

The input size of the relative encoder (green area in Figure 5) is the size of the latent vectors (i.e., the backbone output size) plus the size of point coordinates, i.e., $32+3=35$. All linear layers have an output size of 32, except the multi-head layer for the computation of significance weights, of output size $h=64$, and the final occupancy layer, of output size 2, corresponding to classes empty and full. The layer activations all are ReLUs. Batch norms are only used in the backbone, i.e., the absolute encoder $E$; there are none in the relative encoder $R$, nor in the decoder $D$.

**Point sampling at training time** is not part of POCO. We reused existing dataset samplings (from ConvONet [33] and Points2Surf [14]) to compare on the same training data. The other datasets are only used for inference.

**Training settings.** We train using Adam [26] with learning rate $10^{-3}$. The training batch size is 16 for 3k input points and 8 for 10k input points. We train for 600k iterations.

## B. Meshing for occupancy

Mesh generation, for implicit functions, generally relies on the Marching cubes (MC) algorithm [29], evaluating occupancy on a regular 3D grid.

**Marching cubes based on refinements (MC-refin).** Recently, the MC variant used in ONet [30] has often been used due to its higher speed. It operates on a coarse grid but locally refines the resolution thanks to a heuristics: Unless all corners of a cube at a given resolution agree on being empty of full, i.e., as soon as two corners of a cube disagree on occupancy, the cube is subdivided into 8 subvoxels. The initial grid is typically of size $32^3$, and it is typically refined (subdivided) up to two times, leading to a local resolution equivalent to a $128^3$ grid. The resulting mesh, after MC, is furthermore simplified [16] and refined using first and second order gradient information [30]. While the heuristics may miss thin details, this MC with refinement (MC-refin) leads to a much faster running time than plain MC, with a factor up to $8^2$ when using up to two refinement steps.

**Marching cubes based on region growing (MC-regro).** To ensure we have little chances of missing refinements, in particular for locally complex surfaces or thin volumes, we use a different strategy. We consider from the outset a fine-grained resolution but, to prevent many useless queries in large empty or full regions, we adopt a region-growing

approach (MC-regro). The seeds are the input points, for which we compute the occupancy. We then compute the occupancy for query points that are both in the close neighborhood (voxels at distance at most 2 grid steps) of both a location in the empty volume and a location in the full volume, i.e., close to the surface. And we iterate.

Besides, with the Marching cubes algorithm, a vertex is placed on the edge of a cube by linearly interpolating the two scalar values at the edge's endpoints. But contrary to distance fields, occupancy fields may have sharp transitions. Consequently, opposite-side endpoints frequently have values close to 0 and 1, and vertices tend to be placed in the middle of segments, creating discretization effects. To prevent it, we perform a dichotomic search along edges to better locate the occupancy transition. We operate 10 dichotomies, which is more than enough in most cases.

In general, reconstructions with MC-regro are qualitatively better than MC-refin on scenes, but similar on objects. In fact, quantitative results on ShapeNet show a similar reconstruction accuracy of POCO with either MC-refin or MC-regro. The reason probably is that thin details have little impact on the different metrics. This ability to capture thin details makes MC-regro generally slower than MC-refin (see Section C, Table 7).

## C. Running times

Some running times are given in Figures 1 and 4 in the paper, as well as here in Tables 6 and 7.

The time for the backbone to extract features is negligible ($< 1\%$). The bottleneck is the decoding, as we have to respond to many occupancy queries depending on the resolution of the Marching cubes (MC). And to answer an MC query, the bottleneck is the computation of nearest neighbors, which currently is not optimized, requiring communications between the GPU and the CPU. (It could probably be optimized by pre-computing neighbors at low MC resolution to reduce the GPU-CPU communication overhead.)

In contrast, grid-based methods such as those based on ConvONet [28, 33, 37] do not need such an optimization as they do not depend on nearest neighbors. However, while our approach requires extracting one feature per point for encoding (typically a few thousands points for an object), these other methods extract one feature per grid cell, typically $64^3 \approx 262$k. Besides, as we show in the paper, losing input points induces a loss of details.

**Impact of test-time augmentations.** Although in this case, because of the high point-cloud density (50k pts), we apply the test-time augmentation (TTA) strategy and run the latent vector inference on many different point cloud subsamples (such that each point is seen at least $N_{\text{view}} = 10$ times), our method is still significantly faster than Points2Surf.

| Method and setting | Time |
|---|---|
| Points2Surf | |
| Full reconstruction (single thread) | 23 min 48 s |
| Full reconstruction (1 thread per model) | 10 min 15 s |
| POCO ($N_{\text{train}} = N_{\text{test}} = 3$k, $N_{\text{view}} = 10$) | |
| Only inference of latent vectors | 38 s |
| Full reconstruction (single thread) | 4 min 27 s |

Table 6. **Running time** for reconstructing the 4 models of the Real-World dataset (50k pts each) using Points2Surf or POCO.

| Method | MC-refin | MC-regro | Time |
|---|---|---|---|
| SA-ConvONet | ✓ | | 245.7 s |
| LIG (5k iter.) | ✓ | | 104.5 s |
| LIG (3k iter.) | ✓ | | 66.2 s |
| Points2Surf | ✓ | | 38.4 s |
| SPR | ✓ | | 14.9 s |
| Neural Splines | ✓ | | 12.7 s |
| ConvONet | ✓ | | 0.6 s |
| POCO | | ✓ | 10.7 s |
| POCO | ✓ | | 2.5 s |

Table 7. **Average reconstruction time** of different methods for ShapeNet shapes from 3k points using the same $128^3$ grid size for the Marching cubes (MC), although with different heuristics and MC variants. MC-refin is the commonly-used MC variant in [30] that operates on a $32^3$ grid and potentially refines it locally twice into a local resolution equivalent to a $128^3$ grid. MC-regro is our region-growing variant of the Marching cubes that directly operates on a $128^3$ grid, although sparsely (see Section B).

In fact, as our encoding time is negligible compared to the numerous decoding queries for meshing with MC, our TTA strategy at feature level brings little slowdown, e.g., +5% for $N_{\text{view}} = 10$, compared to $N_{\text{view}} = 1$.

**Overall reconstruction time.** In Table 7, we report the average reconstruction time of different methods. To be fair, given that mesh generation via occupancy queries is a running time bottleneck, we compare the methods using the same MC algorithm, namely MC-refin with a coarse grid of size $32^3$ that can be refined up to twice, i.e., into a grid of size $128^3$. We also report the running time of POCO with our MC-regro variant on a grid of size $128^3$. As said in Section B, the quantitative results of POCO with either MC-refin or MC-regro are similar.

On ShapeNet with medium-density points clouds (3k points per shape), we rank second behind ConvONet for speed. Note however that LIG is faster on denser scenes (see Figure 4 of the main paper) as the computation time per patch is constant, while our kNN search based on a kd-tree gets slower. (It could be faster by precomputing neighbors in the data loader, to limit GPU-CPU exchanges.)

## D. Receptive field

A question that naturally arises to understand the power and benefits of different approaches is the size of the receptive field for inferring occupancy features.

Because it is based on nearest neighbors, the receptive field of the backbone varies based on the scene geometry. It naturally tends to augment with the number of layers but sometimes, as when a separate group of points are mutual neighbors, the local receptive field does not increase.

To evaluate the actual (in fact, maximum) receptive field of a given point, we apply the following procedure:

1. We use a variant of the network without ReLUs and where the convolutions are replaced with averaging.
2. We apply the loss on a single output location.
3. We back-propagate the loss signal.
4. We identify input points receiving a non-zero gradient.

On a SceneNet living-room scene, with density 100 pts/m$^2$, we obtain an average receptive field of 29k points when looking at non-zero gradient (see Figure 7). If we only look at points for which the back-propagated gradient has a norm greater than $10^{-7}$ (i.e., a significant gradient), then the receptive field encompasses 16k points.

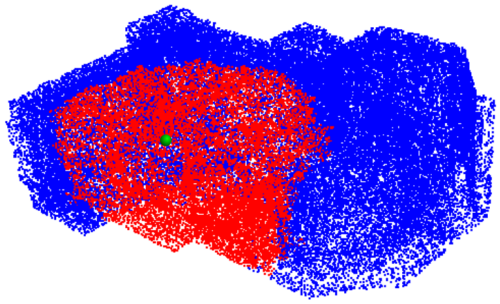


Figure 7. **Receptive field of the FKAConv backbone** on a point cloud from SceneNet with density 100 pts/m$^2$. The receptive field of the point marked in green is colored in red.

## E. Experiments

### E.1. Choice of compared methods and datasets

Following the success of methods such as AtlasNet [19] and DeepSDF [31], a dozen of new learning-based reconstruction methods have been published every year.

As said in the main paper, existing methods often perform well in some settings but not in others. Consequently, most published papers tend to evaluate on different datasets (see Table 9) or in specific configurations: low or high density of train/test points, with or without added noise and outliers, with or without oriented normals, training specifically for a class of shapes or generalizing to any shape, addressing single object or whole scene reconstruction, etc. Some methods are also too slow to be evaluated on full datasets and report results only on dataset fractions. Last, although most methods make code available, some do not offer pretrained models, or scripts, or parameters (at the time of writing). This makes comparisons particularly difficult.

| Train | Test | Train set | Test set |
|---|---|---|---|
| object | object | ShapeNet | ShapeNet |
| object | object | ABC | ABC, *Thingi10k, RealWorld, Famous* |
| object | scene | ShapeNet | *SceneNet* |
| scene | scene | Synthetic Rooms | Synthetic Rooms, *MatterPort3D* |

Table 8. **Datasets used for training and testing.** *Italic:* datasets used in a generalization setting, including from objects to scenes.

We chose to compare to some of the most cited or most recent methods. To be fair with these methods, we evaluate in their setting (when enough information is provided to do so) rather than impose them other specific settings. It also illustrates the ability of our method to adapt to various configurations. Method codes are referenced in Section F.3.

The datasets that we used in our experiments are listed in Table 8. Datasets references are in Section F.3.

- On SceneNet, we chose points with normals, which allows comparing to LIG (which requires normals).
- On MatterPort3D, we chose points without normals, allowing comparison to SA-ConvONet but not to LIG.
- On ShapeNet, we chose in the main paper points without normals and with noise, allowing comparison to ConvONet; in this supplement, we use points with normals and without noise, allowing to compare to LIG.

### E.2. Metrics

We use exactly the same evaluation metrics as ConvONet [33], as specified formally in the supplementary material. However, for our report to be more self-contained, we reformulate here explicitly the metrics that we use.

The surface metrics measure different forms of deviations between two surfaces, i.e., the deviation between the reconstructed surface and the ground-truth surface. In practice, the metrics are approximated by replacing the continuous distances by the distances between points sampled on both surfaces. In particular, the distance of a point $p$ to a surface $S$ is approximated by the distance of $p$ the nearest point $q$ sampled on surface $S$. In our experiments, we sample on each surface: 100k points for ShapeNet and Synthetic Rooms; 10k for ABC, Famous and Thingi10k; and 4M points for SceneNet. As can be seen by the performance of 'Oracle' in Table 5 of the paper, which compares the ground-truth against itself via two different samplings, this discretization is a reasonable approximation, although POCO gets close to the error margin when the point cloud is dense and the normals are provided.

| | | normals required | code unavailable | pre-training unavailable | Objects | | | | | | | Scenes | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 3D Warehouse | ABC | D-Faust | Famous | ShapeNet | Thingi10K | ThreeDScans | 3DFront | MatterPort3D | ScanNet | SceneNet | Synthetic Rooms | Tanks and Temples |
| AdaConv | [38] | ✓ | | ✓ | | | | | | | | | | | | | ■$^{\perp}$ |
| ConvONet | [33] | | | | | | | | ■ | | | | □ | ■ | | ■ | |
| DeepLS | [5] | | ✓ | ✓ | □ | | | | | | | | | | | | |
| DeepSDF | [31] | | | ✓ | | | | | ■ | | | | | | | | |
| DefTet | [15] | | ✓ | ✓ | | | | | ■ | | | | | | | | |
| DP-ConvONet | [28] | | | | | | | | ■ | | | | | | | ■ | |
| IF-NET | [9] | | | | | | | | ■ | | | | | | | | |
| IGR | [18] | | | | | | | | | | | | | | | | |
| IM-NET | [8] | | | | | | | | □ | | | | | | | | |
| LDIF | [17] | | | ✓ | | | | | ■ | | | | | | | | |
| LIG | [24] | ✓ | | | | | | | ■$^{\perp}$ | | | | ■$^{\perp}$ | | ■$^{\perp}$ | | |
| MetaSDF | [36] | | | ✓ | | | | | □ | | | | | | | | |
| NDF | [10] | | | ✓ | | | | | □ | | | | | | | | |
| Neural-Pull | [3] | | | | | □ | | □ | | | | | | | | | |
| Neural Splines | [39] | ✓ | | | | | | | □$^{\perp}$ | | | | | | | | |
| ONet | [30] | | | | | | | | ■$^{\perp}$ | | | | | | | | |
| Points2Surf | [14] | | | | | □ | | □ | | □ | | | | | | | |
| RetrievalFuse | [35] | | | ✓ | | | | | ■ | | | ■* | ■* | | | | |
| SA-ConvONet | [37] | | | | | | | | □ | | | | □ | □ | | □ | |
| SAIL-S3 | [40] | | ✓ | ✓ | | | | | □ | | □ | | | | | | |
| SAL | [1] | | | | | | ■ | | | | | | | | | | |
| SALD | [2] | | ✓ | ✓ | | | ■ | | | | | | | | | | |
| SAP | [32] | | | | | | | | ■ | | | | | | | | |
| ScanComplete | [13] | | | | | | | | | | | | | ■* | | | |
| SG-NN | [12] | | | | | | | | | | | | ■* | | | | |
| SPR | [25] | ✓ | | | | | | | | | | | | | | | |
| POCO | (ours) | | | | | □ | | □ | ■ | □ | | | □ | | ■$^{\perp}$ | ■ | |

Table 9. **Datasets used for the *evaluation* of 3D reconstruction methods *from point clouds* in their published paper**, if freely available and $>$ *10 shapes are used*, and availability of code or pre-trained models suited for testing on the datasets (at the time of writing).

■: test on all/many shapes of the dataset ($>$ 1000), □: test on a few shapes ($\leq$ 100 or a single category), $^{\perp}$: test with ground-truth normals as input, *: actual scans rather than uniformly sampled points.

Tests on a given dataset may however be done in different settings (number of sampled points, amount of added noise or outliers, use many shapes but excluded classes or objects, etc.). For instance, many different numbers can be found in various publications for the performance of ONet on the ShapeNet dataset.

**Chamfer distance (CD).** The Chamfer distance between two point clouds $P_1, P_2$ is defined as follows:

$$\text{Chamfer}(P_1, P_2) = \frac{1}{2\,|P1|} \sum_{p_1 \in P_1} \min_{p_2 \in P_2} d(p_1, p_2) + \frac{1}{2\,|P2|} \sum_{p_2 \in P_2} \min_{p_1 \in P_1} d(p_1, p_2)$$

where $d(p_1, p_2)$ is the distance between points $p_1, p_2$. In the paper, following ONet [30] and ConvONet [33], we use the L1-norm. What we name 'CD' in tables is Chamfer $\times\, 10^2$.

**Normal consistency (NC).** The normal consistency between two point clouds $P_1, P_2$ is defined as follow:

$$\text{NC}(P_1, P_2) = \frac{1}{2\,|P1|} \sum_{p_1 \in P_1} n_{p_1} . n_{\text{closest}(p_1, P_2)} + \frac{1}{2\,|P2|} \sum_{p_2 \in P_2} n_{p_2} . n_{\text{closest}(p_2, P_1)}$$

where

$$\text{closest}(p, P) = \arg\min_{p' \in P} d(p, p')$$

is the closest point to $p$ in point cloud $P$ and where $n_p$ is the normal at point $p$, given by the orientation of the mesh face on which the point is sampled.

**F-Score (FS).** The F-Score between two point clouds $P_1$ and $P_2$ at a given threshold $t$ is given by:

$$\text{FS}(t, P_1, P_2) = \frac{2\,\text{Recall}\;\text{Precision}}{\text{Recall} + \text{Precision}}$$

where

$$\text{Recall}(t, P_1, P_2) = \left| \left\{ p_1 \in P_1,\ \text{s.t.} \min_{p_2 \in P_2} d(p_1, p_2) < t \right\} \right|$$
$$\text{Precision}(t, P_1, P_2) = \left| \left\{ p_2 \in P_2,\ \text{s.t.} \min_{p_1 \in P_1} d(p_2, p_1) < t \right\} \right|$$

In the paper, following ONet [30] and ConvONet [33], we use $t = 0.01$.

**Intersection over Union (IoU).** Compared to the previous metrics, which evaluates the quality of the generated surface, the IoU is a volume metric.

Noting TP (resp. FP and FN) the number of true positive, i.e., the number of points correctly predicted as full (resp. the number of points wrongly predicted as full, and the number of points wrongly predicted as empty), the IoU is defined as follows:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

### E.3. More qualitative results

**POCO vs LIG on ShapeNet (various densities).** We provide on Figure 8 more visualizations of ShapeNet reconstructions, comparing LIG to POCO at various densities of input points (with normals). LIG reconstructions were done using the best parameter setting for the method, i.e., with part size 0.20 for 512 and 2048 points, and part size 0.10 for 8192 points. Nevertheless, POCO reconstructs surfaces with more robustness and much sharper details.

**POCO vs SPR and LIG on SceneNet (various densities).** As a complement to Table 5 in the main paper, we provide here on Figure 9 the visualization of a reconstruction fragment of a SceneNet scene, also with varying input point densities, comparing SPR, LIG and POCO. As can be seen, POCO provides a better robustness at low point densities and more details at high point densities.

**POCO vs SPR (generalization ability).** In fact, POCO out-of-the-box adapts well to new shape domains without retraining (Figures 1, 3, 4 and Table 2), especially when given normals (Table 5). SPR only works well on high-density point clouds (Figure 4, Tables 2, 4, 5).

**POCO vs ConvOnet on Synthetic Rooms.** As a complement to Table 4 in the main paper, we provide here on Figure 10 the visualization of reconstructions on the SyntheticRooms dataset (2 first scenes of each data bunch), comparing ConvOnet and POCO. In general, we provide more and sharper details; we are also more robust to thin surfaces, e.g., selves of the bookcase in "Room 05 - scene 801" and coffee table in the foreground of "Room 08 - scene 801".

### E.4. More quantitative results

**POCO vs PointConv, ONet and ConvOnet on ShapeNet.** As a complement to Table 3 in the main paper, we provide here in Table 10 classwise quantitative results on ShapeNet, comparing POCO to PointConv, ONet and ConvONet (the $3 \times 64^2$ variant, that performs best on ShapeNet).

PointConv is a baseline method which is defined in the ConvONet paper [33]. It proceeds as follows: point-wise features are extracted using PointNet++ [34], interpolated using Gaussian kernel regression and feed into the same fully-connected network used in ConvONet [33]. While this baseline uses local information, it does not exploit convolutions. ONet [30] is not convolutional either; it operates on shapes as a whole.

As can be seen in the table, POCO largely outperforms the compared methods on all categories, especially on classes featuring complex details such as *lamp*, *rifle*, *vessel* and, to a lesser extent, *airplane*, *car*, *chair* and *loudspeaker*. Yet, the most difficult classes are more or less the same for all methods, including POCO: *lamp* and *car*.

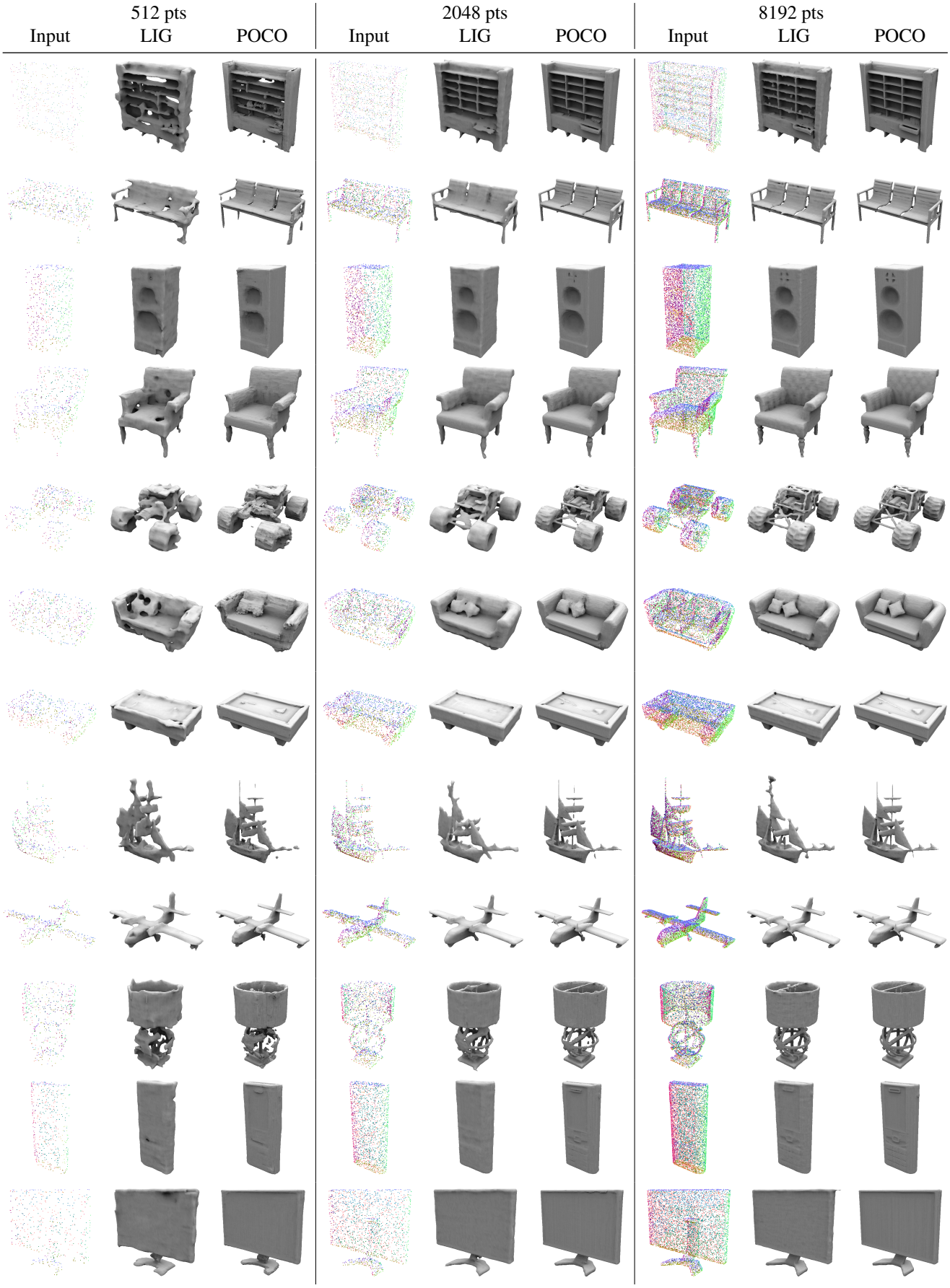



Figure 8. **ShapeNet reconstructions** (input with normals), LIG (part size 0.20 for 512 and 2048 pts, 0.10 for 8192 pts) and POCO (ours).

Figure 9. **Reconstruction fragment of a SceneNet scene** with varying input point densities for SPR, LIG and POCO.

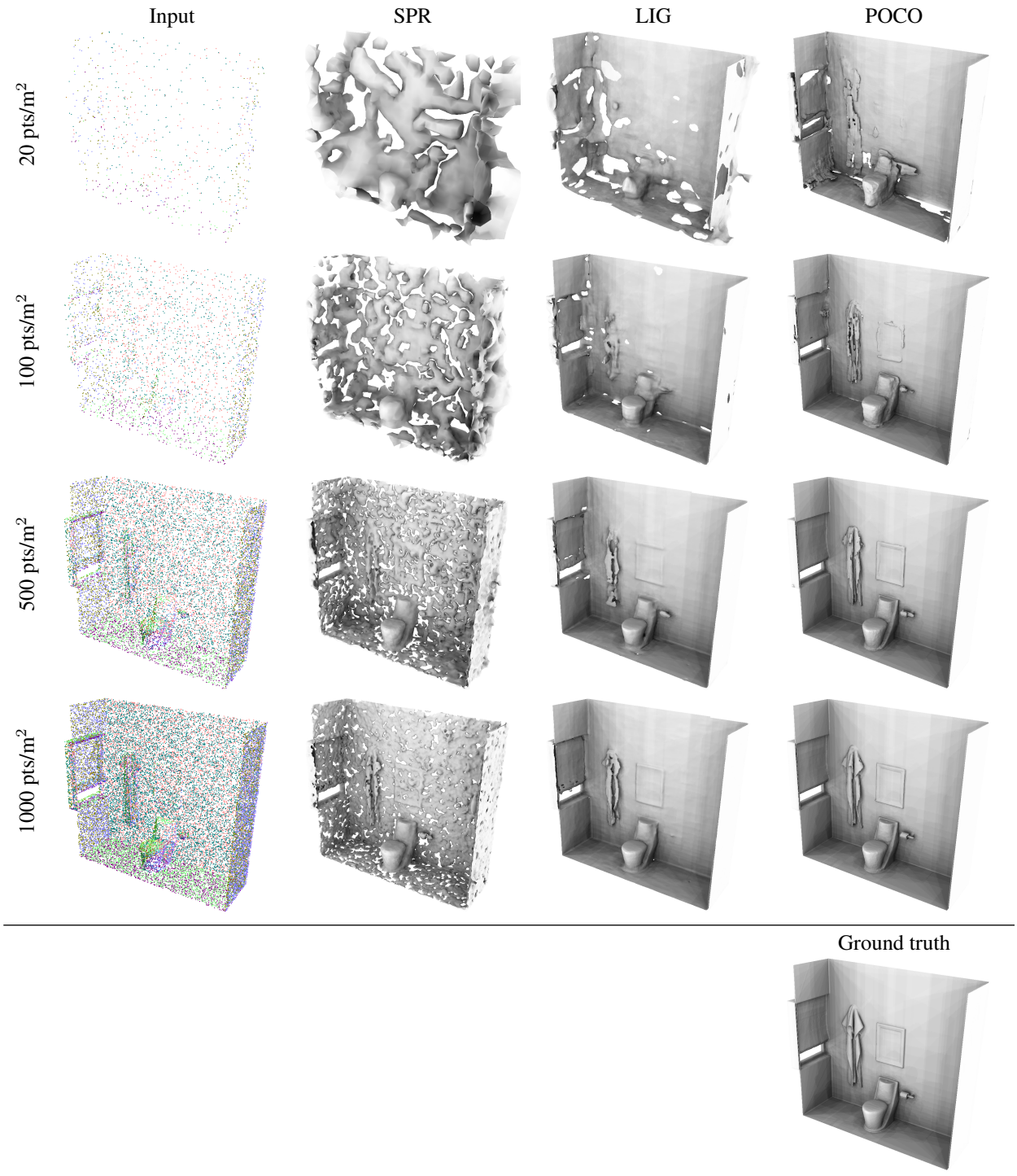
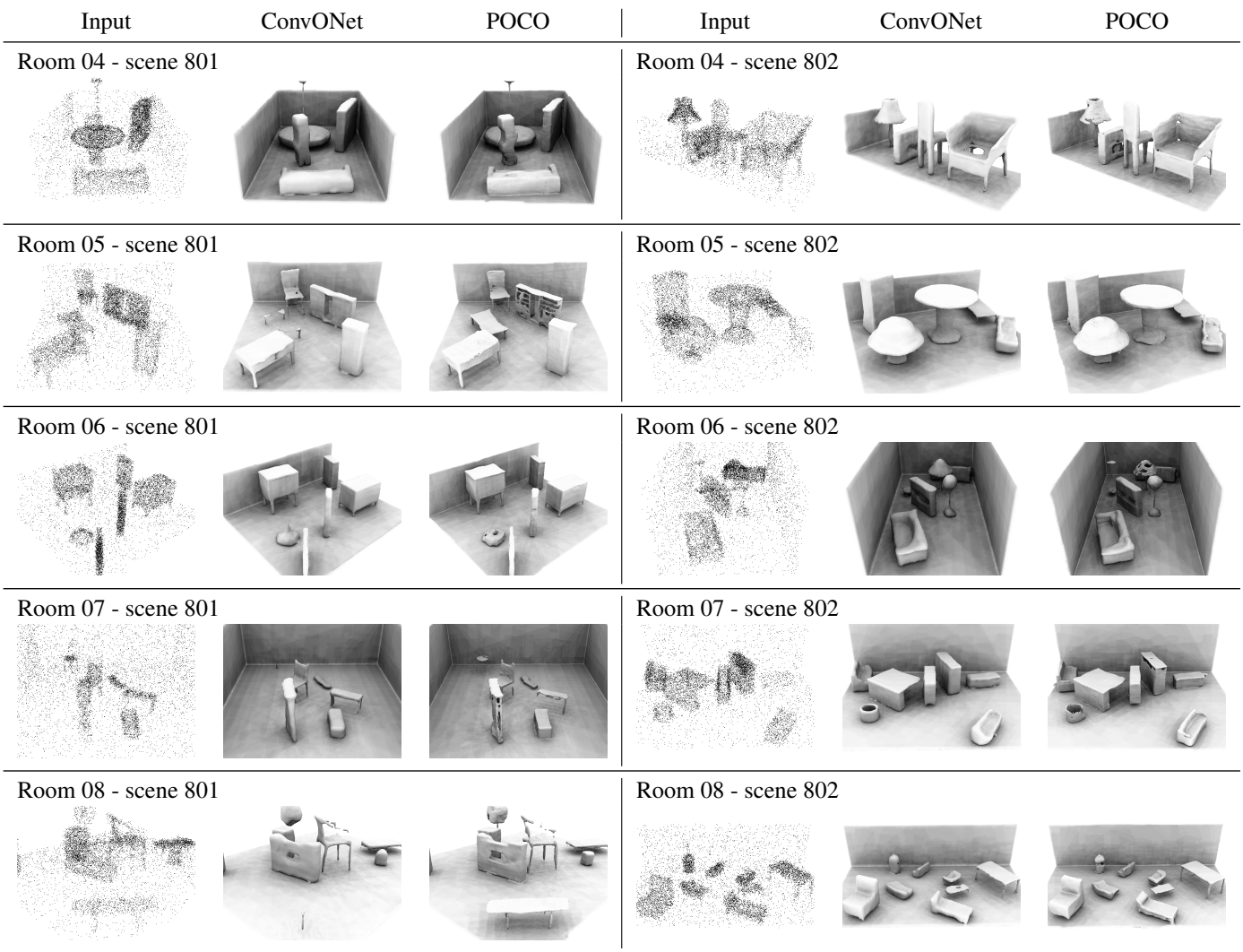


Figure 10. **Synthetic Rooms** reconstructions using ConvONet and POCO (ours), from 10k points with noise.



| | IoU ↑ | | | | CD↓ | | | |
|---|---|---|---|---|---|---|---|---|
| Category | PointConv | ONet | ConvONet | POCO | PointConv | ONet | ConvONet | POCO |
| Airplane | 0.579 | 0.734 | 0.849 | **0.902** | 1.40 | 0.64 | 0.34 | **0.23** |
| Bench | 0.537 | 0.682 | 0.830 | **0.865** | 1.20 | 0.67 | 0.35 | **0.28** |
| Cabinet | 0.824 | 0.855 | 0.940 | **0.960** | 1.15 | 0.82 | 0.46 | **0.37** |
| Car | 0.767 | 0.830 | 0.886 | **0.921** | 1.49 | 1.04 | 0.75 | **0.41** |
| Chair | 0.667 | 0.720 | 0.871 | **0.919** | 1.29 | 0.95 | 0.46 | **0.33** |
| Display | 0.743 | 0.799 | 0.927 | **0.956** | 1.06 | 0.82 | 0.36 | **0.28** |
| Lamp | 0.495 | 0.546 | 0.785 | **0.877** | 2.15 | 1.59 | 0.59 | **0.33** |
| Loudspeaker | 0.807 | 0.826 | 0.918 | **0.957** | 1.48 | 1.18 | 0.64 | **0.41** |
| Rifle | 0.565 | 0.668 | 0.846 | **0.897** | 0.98 | 0.66 | 0.28 | **0.19** |
| Sofa | 0.811 | 0.865 | 0.936 | **0.963** | 1.04 | 0.73 | 0.42 | **0.30** |
| Table | 0.654 | 0.739 | 0.888 | **0.924** | 1.13 | 0.76 | 0.38 | **0.31** |
| Telephone | 0.856 | 0.896 | 0.955 | **0.968** | 0.61 | 0.46 | 0.27 | **0.22** |
| Vessel | 0.652 | 0.729 | 0.865 | **0.927** | 1.38 | 0.94 | 0.43 | **0.25** |
| Mean | 0.689 | 0.761 | 0.884 | **0.926** | 1.26 | 0.87 | 0.44 | **0.30** |

| | NC ↑ | | | | FS↑ | | | |
|---|---|---|---|---|---|---|---|---|
| Category | PointConv | ONet | ConvONet | POCO | PointConv | ONet | ConvONet | POCO |
| Airplane | 0.819 | 0.886 | 0.931 | **0.944** | 0.562 | 0.829 | 0.965 | **0.994** |
| Bench | 0.811 | 0.871 | 0.921 | **0.928** | 0.617 | 0.827 | 0.964 | **0.988** |
| Cabinet | 0.895 | 0.913 | 0.956 | **0.961** | 0.719 | 0.833 | 0.956 | **0.979** |
| Car | 0.845 | 0.874 | 0.893 | **0.894** | 0.577 | 0.747 | 0.849 | **0.946** |
| Chair | 0.851 | 0.886 | 0.943 | **0.956** | 0.618 | 0.730 | 0.939 | **0.985** |
| Display | 0.910 | 0.926 | 0.968 | **0.975** | 0.679 | 0.795 | 0.971 | **0.994** |
| Lamp | 0.779 | 0.809 | 0.900 | **0.929** | 0.453 | 0.581 | 0.892 | **0.975** |
| Loudspeaker | 0.894 | 0.903 | 0.939 | **0.952** | 0.647 | 0.727 | 0.892 | **0.964** |
| Rifle | 0.796 | 0.849 | 0.929 | **0.949** | 0.682 | 0.818 | 0.980 | **0.998** |
| Sofa | 0.900 | 0.928 | 0.958 | **0.967** | 0.697 | 0.832 | 0.953 | **0.989** |
| Table | 0.878 | 0.917 | 0.959 | **0.966** | 0.694 | 0.824 | 0.967 | **0.991** |
| Telephone | 0.961 | 0.970 | 0.983 | **0.985** | 0.880 | 0.930 | 0.989 | **0.998** |
| Vessel | 0.817 | 0.857 | 0.919 | **0.940** | 0.550 | 0.734 | 0.931 | **0.989** |
| Mean | 0.858 | 0.891 | 0.938 | **0.950** | 0.644 | 0.785 | 0.942 | **0.984** |

Table 10. **Classwise ShapeNet reconstruction.** All models are trained on 3k noisy points. Results for methods other than POCO are reported from the supplementary material of ConvONet [33].

# F. Use of existing assets

## F.1. Pre-existing code

The implementation of our approach has several dependencies, that are all free to use for research purposes. The main dependencies of our code are as follows:

- **FKAConv**[1] [4], under Apache License v2.0.
- **PyTorch**[2], under the Apache CLA,
- **PyTorch-Geometric**[3], under the MIT License,

The code of POCO[4] itself is freely available, under Apache License v2.0.

## F.2. Datasets

For the experiments, we used several datasets that are freely available for research purpose:

- **ABC**[5] is under the Onshape Terms of Use[6]. We used the subset preprocessed and made available by the authors of Points2Surf[18] [14].
- **Famous** is a set of shapes of various origins, among which the Stanford 3D Scanning Repository[7] [27]. This set of shapes is described, preprocessed and made available by the authors of Points2Surf[18] [14].
- **MatterPort3D**[8] [6] is under a user license agreement for academic use. We used scenes preprocessed by the authors of SA-ConvONet[19] [37].
- **Real-World** point clouds used in the paper are described, preprocessed and made available by the authors of Points2Surf[18] [14].
- **SceneNet**[9] [20–22] is under the CC BY-NC 4.0, for research purposes only. We made meshes watertight using **Watertight Manifold**[10] [23], that enables code use under mild conditions.
- **ShapeNet**[11] [7] has a licence for non commercial research or educational purposes. We used the version of ShapeNet as preprocessed by the authors of **ONet**[12] [30], which itself reuses the preprocessing of the authors of **3D-R2N2**[13] [11].
- **Synthetic Rooms**[15] is a dataset created by the authors of ConvONet [33] based on ShapeNet models.
- **Thingi10K**[14] [41] is a freely available collection of shapes under various licences. We used the subset preprocessed and made available by the authors of Points2Surf[18] [14].

## F.3. Methods

We compared to a number of reconstruction methods, reusing the code made available by their authors:

- **ConvONet**[15] [33] under the MIT License.
- **LIG**[16] [24] probably under Apache License v2,
- **Neural Splines**[17] [39] under the MIT License,
- **Points2Surf**[18] [14] under the MIT License,
- **SA-ConvONet**[19] [37] under the MIT License,
- **SPR**[20] [25] under the MIT License.

We also compared to **AtlasNet** [19], **DeepSDF** [31], **DP-ConvONet** [28], **ONet** [30], but only reusing the numbers mentioned in [28, 33].

Here are some methods we would have liked to compare to, but could not in practice:

- **AdaConv**[21] [38]: The repository provides raw code but no pre-trained model nor instructions or scripts to train or to test, which may lead to misuses and wrong comparisons.
- **NDF**[22] [10]: The repository provides code but only a pre-trained model for ShapeNet cars. For scene reconstruction, it does not offer preprocessed data or any data preprocessing procedure to retrain a model, nor instructions to run NDF using a sliding window scheme, as alluded to in the supplementary material.

As indicated in Table 9, some authors also have not made their code or their model available to allow comparisons.

# G. Societal impact

We believe our 3D reconstruction approach has **very little potential for malicious uses** (including disinformation, surveillance, invasion of privacy, endangering security), not

[1] https://github.com/valeoai/FKAConv
[2] https://pytorch.org/
[3] https://pytorch-geometric.readthedocs.io/
[4] https://github.com/valeoai/POCO
[5] https://deep-geometry.github.io/abc-dataset/
[6] https://www.onshape.com/en/legal/terms-of-use
[7] http://graphics.stanford.edu/data/3Dscanrep/
[8] https://niessner.github.io/Matterport/
[9] https://robotvault.bitbucket.io/
[10] https://github.com/hjwdzh/Manifold
[11] https://shapenet.org/
[12] https://github.com/autonomousvision/occupancy_networks
[13] https://github.com/chrischoy/3D-R2N2
[14] https://ten-thousand-models.appspot.com
[15] https://github.com/autonomousvision/convolutional_occupancy_networks
[16] https://github.com/tensorflow/graphics/tree/master/tensorflow_graphics/projects/local_implicit_grid
[17] https://github.com/fwilliams/neural-splines
[18] https://github.com/ErlerPhilipp/points2surf
[19] https://github.com/tangjiapeng/SA-ConvONet
[20] https://github.com/mkazhdan/PoissonRecon
[21] https://github.com/isl-org/adaptive-surface-reconstruction
[22] https://github.com/jchibane/ndf

more, e.g., than image enhancement methods in the 2D data case, and not more than hundreds of previously published 3D reconstruction methods. Besides, we are not bound nor promoting any dataset that would lead to unfairness in any sense. The use of our method has a **modest environmental impact** as the training time (a few days on a single GPU for a large dataset) and the inference times (minutes, or hours for very large point clouds) are somewhat moderate, and favorably compare to many learning-based approaches.

On the contrary, applications of our method can be found in various domains, with positive societal impacts:

**Heritage preservation.** Digitizing cultural objects and monuments allows a form of heritage preservation and enables virtual museums to make works of art and culture more widely accessible.

**Infrastructure and building maintenance.** Reconstructing models of existing infrastructures and buildings is of high interest for the construction industry. These models are particularly useful to plan and organize maintenance. This is particularly useful in a context of aging infrastructures and building renovation for energy-saving insulation.

**Augmented and virtual reality.** Surface and volume reconstruction are useful assets for augmented and virtual reality, whether it is for professional use (e.g., on-site maintenance of equipment) or entertainment (video games, special effects for the film industry), which is however to be consumed in moderation.

## References


[1] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4

[2] Matan Atzmon and Yaron Lipman. SALD: Sign agnostic learning with derivatives. In *International Conference on Learning Representations (ICLR)*, 2021. 4

[3] Ma Baorui, Han Zhizhong, Liu Yu-shen, and Zwicker Matthias. Neural-Pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. In *International Conference on Machine Learning (ICML)*, 2021. 4

[4] Alexandre Boulch, Gilles Puy, and Renaud Marlet. FKAConv: Feature-kernel alignment for point cloud convolution. In *Asian Conference on Computer Vision (ACCV)*, 2020. 1, 10

[5] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, S. Lovegrove, and Richard A. Newcombe. Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. In *European Conference on Computer Vision (ECCV)*, 2020. 4

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision (3DV)*, pages 667–676. IEEE, 2017. 10

[7] A.X. Chang, T.A. Funkhouser, L.J. 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, 2015. arXiv preprint arXiv:1512.03012. 10

[8] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 4

[9] 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. 4

[10] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 4, 10

[11] 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 (ECCV)*, 2016. 10

[12] Angela Dai, Christian Diller, and Matthias 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. 4

[13] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. ScanComplete: Large-scale scene completion and semantic segmentation for 3D scans. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4

[14] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J. Mitra, and Michael Wimmer. Points2Surf: Learning implicit surfaces from point clouds. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 4, 10

[15] Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3D reconstruction. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 4

[16] M. Garland and P.S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Visualization*, pages 263–269, 1998. 1

[17] K. Genova, F. Cole, A. Sud, A. Sarna, and T.A. Funkhouser. Local deep implicit functions for 3D shape. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 4

[18] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning (ICML)*, 2020. 4

[19] T. Groueix, M. Fisher, V.G. Kim, B.C. Russell, and M. Aubry. AtlasNet: A papier-mâché approach to learning 3D surface generation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 10

[20] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. SceneNet: Understanding real world indoor scenes with synthetic data, 2015. preprint arXiv:1511.07041. 10

[21] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Understanding real world

indoor scenes with synthetic data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4077–4085, 2016. 10

[22] Ankur Handa, Viorica Patraucean, Simon Stent, and Roberto Cipolla. SceneNet: An annotated model generator for indoor scene understanding. In *International Conference on Robotics and Automation (ICRA)*, 2016. 10

[23] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for ShapeNet models. *arXiv preprint arXiv:1802.01698*, 2018. 10

[24] C. 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. 1, 4, 10

[25] M.M. Kazhdan and H. Hoppe. Screened Poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3), 2013. 4, 10

[26] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Machine Learning (ICML)*, 2015. 1

[27] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *ACM Conference on Computer Graphics and Interactive Techniques (PACM CGIT)*, pages 313–324, 1996. 10

[28] Stefan Lionar, Daniil Emtsev, Dusan Svilarkovic, and Songyou Peng. Dynamic plane convolutional occupancy networks. In *Winter Conference on Applications of Computer Vision (WACV)*, 2021. 2, 4, 10

[29] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics (CG)*, 21(4):163–169, 1987. 1

[30] 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. 1, 2, 4, 5, 10

[31] J.J. Park, P. Florence, J. Straub, R.A. Newcombe, and S. Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 4, 10

[32] Songyou Peng, Chiyu "Max" Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape As Points: A differentiable Poisson solver. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 4

[33] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3, 4, 5, 9, 10

[34] C.R. Qi, L. Yi, H. Su, and L.J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 5

[35] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. RetrievalFuse: Neural 3D scene reconstruction with a database. In *International Conference on Computer Vision (ICCV)*, 2021. 4

[36] Vincent Sitzmann, Eric R. Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-learning signed distance functions. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 4

[37] Jiapeng Tang, Jiabao Lei, Dan Xu, Feiying Ma, Kui Jia, and Lei Zhang. SA-ConvONet: Sign-agnostic optimization of convolutional occupancy networks. In *International Conference on Computer Vision (ICCV)*, 2021. 2, 4, 10

[38] Benjamin Ummenhofer and Vladlen Koltun. Adaptive surface reconstruction with multiscale convolutional kernels. In *International Conference on Computer Vision (ICCV)*, 2021. 4, 10

[39] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural splines: Fitting 3D surfaces with infinitely-wide neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 4, 10

[40] Wenbin Zhao, Jiabao Lei, Yuxin Wen, Jianguo Zhang, and Kui 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. 4

[41] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3D-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 10