# OffsetOPT: Explicit Surface Reconstruction without Normals

Supplementary Material

## 8. Additional ablation study

Table 7 presents the comparison of reconstruction accuracy for different offset initializations discussed in §4.3.

**Voxel sizes.** We examine how voxel size affects the reconstruction accuracy using 100 shapes from Thingi10k [51]. Each input point cloud consists of 0.1 million randomly sampled points. We normalize the points to fit within a unit sphere, and reconstruct each sample with OffsetOPT using voxel sizes ranging from 0.01 to 0.05. We plot in Fig. 6(left) the variations in CD1 and F1 score to show how the reconstruction accuracy drops as voxel size increases.

**Data noise.** Using the same Thingi10k point clouds, we further evaluate the effect of data noise on OffsetOPT. Specifically, we add varying levels of Gaussian noise (*i.e.*, $\sigma \in \{0, 0.001, 0.002, 0.003, 0.005, 0.1\}$) to create noisy point clouds. Figure 6(right) illustrates our performance as noise increases in the shapes. We note that computational reconstruction methods use noisy points directly as mesh vertices in the reconstructed surface, resulting in reduced robustness to noise compared to implicit methods.

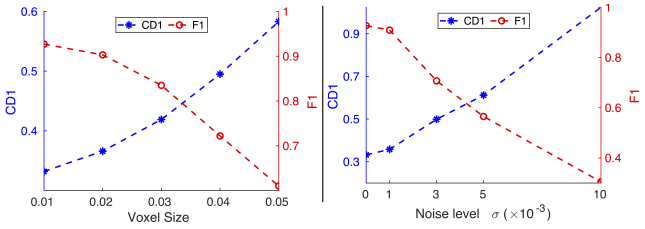



Figure 6. Ablation studies. (left) Effects of voxel sizes on reconstruction quality. (right) Method robustness to data noise.

**KNN influence.** We always sort the KNN points by distances (§3.1). In the training set, since distant points receive the label 0 in the ground-truth matrices $\mathbf{O}^*$, the model learns to discourage problematic edges or triangles itself in stage one. To illustrate, we train another model using $K$=100 and test it on FAUST and MGN. The results are similar to those of $K$=50, shown in Table 6. We suggest 50 for efficiency.

Table 6. Different $K$ in KNN for reconstruction accuracy.

| KNN | FAUST | | | | | MGN | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | CD1↓ | CD2↓ | F1↑ | NC↑ | NR↓ | CD1↓ | CD2↓ | F1↑ | NC↑ | NR↓ |
| $K$=50 | 0.217 | 0.301 | 0.996 | 0.985 | 4.038 | 0.278 | **0.511** | 0.964 | 0.991 | 2.967 |
| $K$=100 | 0.217 | 0.301 | 0.996 | 0.985 | **3.802** | 0.278 | 0.512 | 0.964 | 0.991 | **2.803** |

## 9. Large-scale Reconstruction

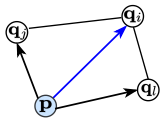
**Chunk processing.** The OffsetOPT approach updates per-point offsets using gradients computed from the entire input point cloud. In large-scale reconstructions, where memory limitations prevent processing all data at once, we divide the input into manageable chunks. Gradients from each chunk are accumulated, and offsets are updated after such accumulation. This chunking strategy is flexible, allowing random splits based on point number and adapting to GPU memory capacity, without the need for careful sub-volume division. Algorithm 2 shows our pseudocode for such processing in PyTorch. It is worth noting that, for convenience, we normalize every input point cloud into a unit sphere in all reconstructions, regardless of the original data scale.

**Algorithm 2** Offset Gradient Accumulation in Pytorch.

**Input:** Number of points per-chunk $I$.
**Output:** Raw gradients of offsets.

```
Compute the number of chunks C = ⌈N/I⌉.
optimizer.zero_grad().
for iteration c < C do
    Get chunk points indexed from I×c to I×(c+1).
    Forward pass their KNN inputs to get chunk_logits.
    chunk_loss = BCE(chunk_logits).
    Accumulate gradients with chunk_loss.backward().
end for
return offsets.grad.
```

**Angle between adjacent triangles.** We compute the angle between two triangles sharing an edge in §3.1 as follows. Let the two triangles be $(\mathbf{p}, \mathbf{q}_i, \mathbf{q}_j)$ and $(\mathbf{p}, \mathbf{q}_i, \mathbf{q}_l)$, $i \neq j \neq l$. We compute the angle $A$ between them as




$$\mathbf{N}_j = (\mathbf{q}_i - \mathbf{p}) \times (\mathbf{q}_j - \mathbf{p}), \tag{9}$$

$$\mathbf{N}_l = (\mathbf{q}_i - \mathbf{p}) \times (\mathbf{q}_l - \mathbf{p}), \tag{10}$$

$$A = \arccos\left(\left\langle \frac{\mathbf{N}_j}{\|\mathbf{N}_j\|}, \frac{\mathbf{N}_l}{\|\mathbf{N}_l\|} \right\rangle\right). \tag{11}$$

For planar surfaces, the angle between any two adjacent triangles sharing an edge will be 180°.

## 10. More Visualizations

Figure 8 compares the reconstructed scene surfaces of SPSR, NKSR, and the proposed OffsetOPT, with triangulation details. Consistently, we observe that SPSR produces surfaces with undesired over-smoothing, while NKSR requires ground-truth normals to perform comparably to our method. Figure 7 shows our reconstructed shape surfaces with wireframes for ABC, FAUST, MGN, and Thingi10k, demonstrating the satisfactory triangulation capability of OffsetOPT. Figure 8 provides further comparison of these methods on large-scale scene reconstruction.

Table 7. Reconstruction accuracy of different offset initializations.

| Dataset | Initialization | Surface Quality | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | overall | | | | | sharp | |
| | | CD1($\times 10^2$)↓ | CD2($\times 10^5$)↓ | F1↑ | NC↑ | NR↓ | ECD1($\times 10^2$)↓ | EF1↑ |
| FAUST | Proposed | 0.217 | 0.301 | 0.996 | 0.985 | 4.038 | **0.561** | **0.896** |
| | zero | 0.217 | 0.301 | 0.996 | 0.985 | **3.835** | 0.584 | 0.889 |
| Thingi10k | Proposed | 0.332 | **0.699** | **0.927** | **0.984** | **5.523** | **4.252** | **0.478** |
| | zero | 0.332 | 0.701 | 0.926 | 0.983 | 5.534 | 4.484 | 0.476 |

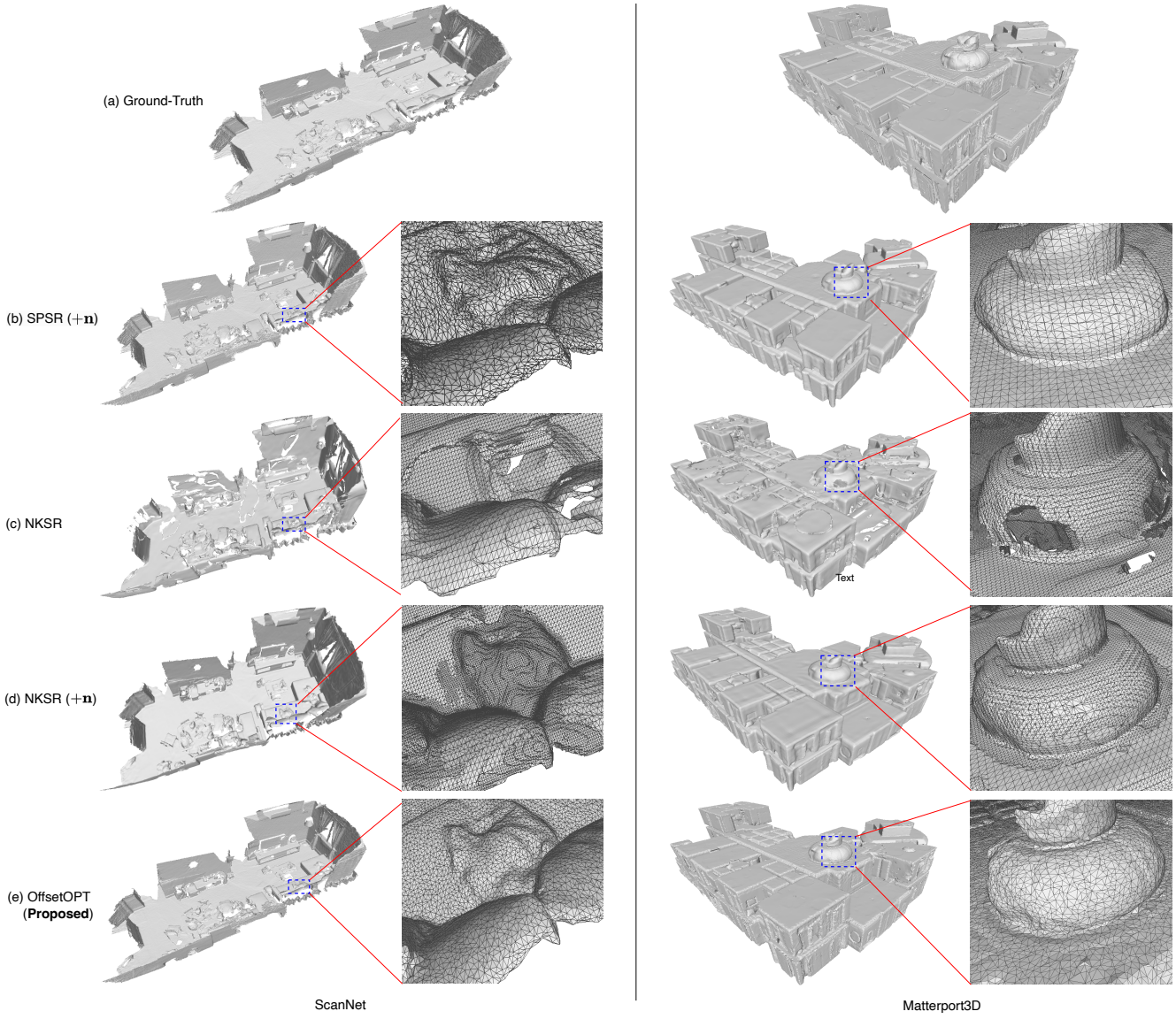



Table 8. Reconstructed surfaces of SPSR, NKSR, and OffsetOPT, with triangulation details for a scene from ScanNet and a scene from Matterport3D. Our method recovers surfaces with sharp features, while NKSR requires ground-truth normals to achieve comparable quality.

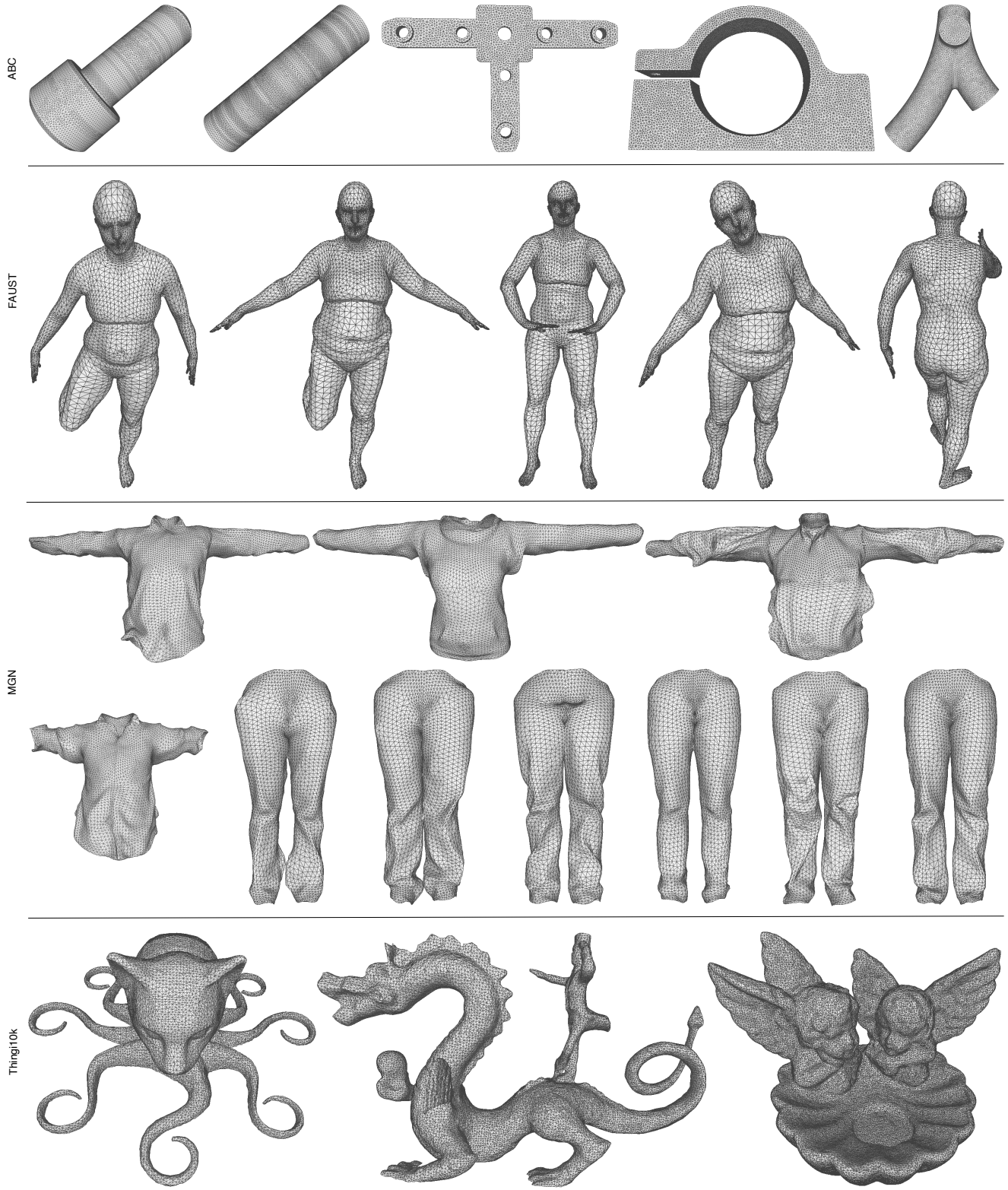



Figure 7. Examples of reconstructed shape surfaces (with wireframes) from ABC, FAUST, MGN, and Thingi10k using the proposed OffsetOPT. For the ABC dataset, results are obtained with a single forward pass of the trained network, without offset optimization, as mentioned in the main paper. OffsetOPT demonstrates surface reconstruction with satisfactory triangulations for all these shape datasets.

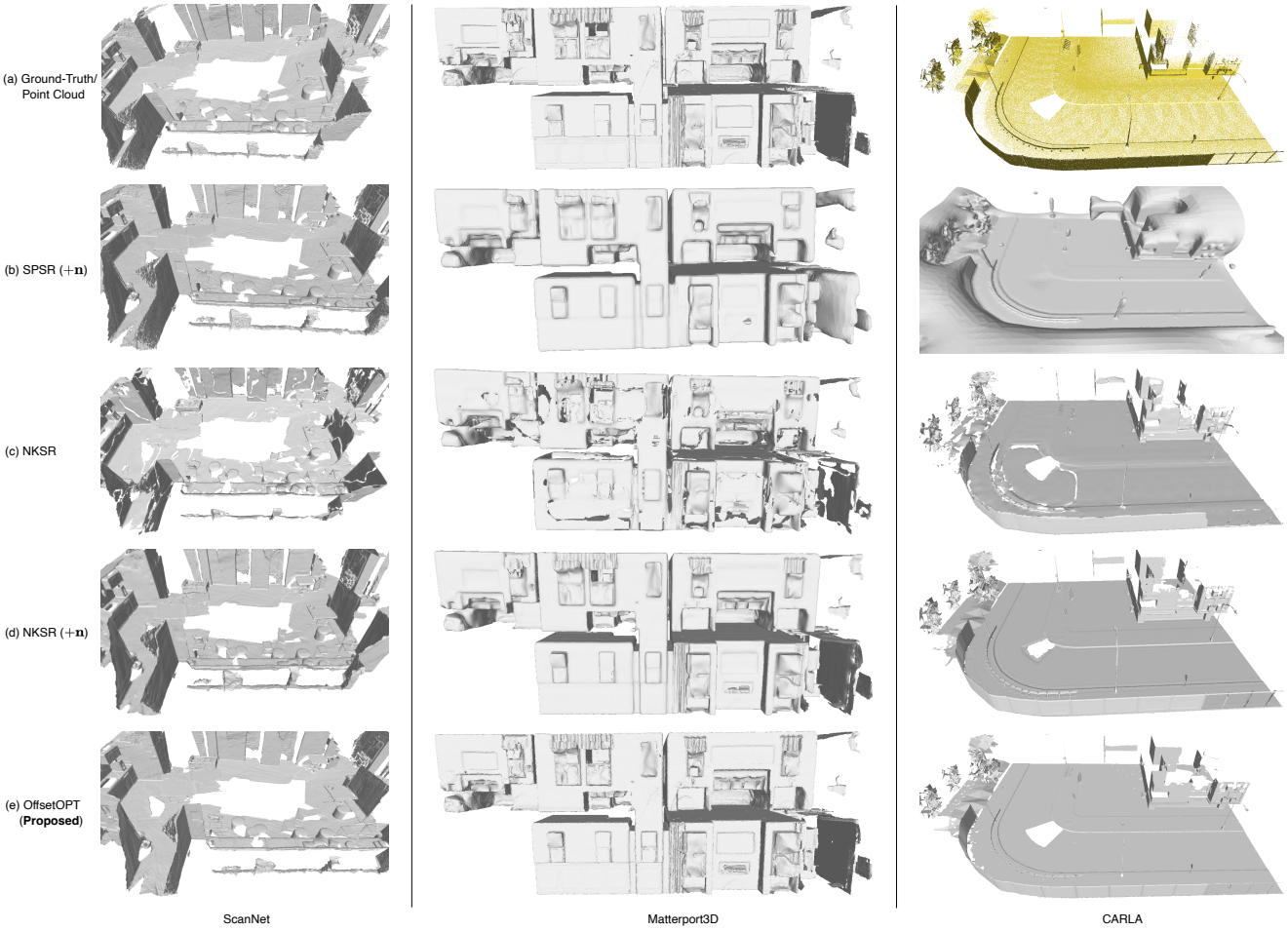



Figure 8. More comparison of different methods on reconstructing the large-scale scenes from ScanNet, Matterport3D, and CARLA.