## A. Overview

In the appendix, we provide the detailed algorithms and additional discussions for computing the parallel $N$-direction frame fields, the mapping of neighborhood patches onto tangent planes and the resampling of feature maps there. The network structures and training details of experiments in the text, as well as additional results and comparisons with previous methods are also presented.

## B. Computing the parallel frame fields

Given a 3D surface mesh, the smooth or parallel frame field that approximates parallel transport of tangent spaces for neighboring points can be efficiently constructed [45]. In particular, we adopt the complex number based approach [21, 8] to encode the $N$-direction fields. We identify the tangent plane $T_xM$ with the complex plane, and a set of unit length vectors $\{u \cdot e^{ik\frac{2\pi}{N}} | k = 0, \cdots, N-1\} \subset \mathbb{C}$ forming a rotationally symmetric $N$-direction frame can be conveniently encoded by their common $N$-th order power $z = u^N \in \mathbb{C}$. To compute a smooth frame field that a) deviates from the parallel transport minimally and b) aligns with salient geometric features of the domain surface, we solve the following optimization problem:

$$\min_{\{z_i\}} \sum_{i \sim j} \|z_i - t_{ji} z_j\|^2 + \lambda \sum_i w_i \|z_i - z_i^0\|^2, \quad (3)$$

where $i, j$ are neighboring vertices on the surface mesh, $t_{ji} \in \mathbb{C}$ is the discrete parallel transport along the edge $ij$ that rotates the tangent plane of $j$ to identify with that of $i$ [21], $\lambda$ is the weight for the second curvature direction alignment term, $w_i = \tanh(|k_{max} - k_{min}|)$ measures the anisotropy at the $i$-th vertex using its maximum and minimum principle curvature values $k_{max}, k_{min}$, and $z_i^0$ is the complex $N$-th order power of the maximum curvature direction at the vertex. The first term is a discretization of the Dirichlet energy of the frame field that measures its variation and encourages parallelism. The second term encourages alignment of the frame field to strong anisotropic directions and salient geometric features of the surface.

As shown in Fig. 8, the smooth frame fields aligned with salient geometric features show strong consistency among deformed shapes, and the singular points are placed consistently at regions with high curvature.

**Alignment to anisotropy.** We test how different balances of field smoothness and alignment to strong anisotropy of the surfaces affect performances. We generate four different sets of frames for the registration task, using $\lambda = 0, 0.01, 0.1, 1$ respectively. The testing accuracies are reported in Table 7, where "SF", meaning smooth frames without curvature direction alignment, corresponds to $\lambda = 0$ and $\|z\| = 1$ to prevent degenerate solutions. From the results, we see that a mild alignment to strong surface anisotropic directions is helpful in achieving the best performances. Therefore, we have used $\lambda = 0.01$ for all tasks shown in other parts of the paper.

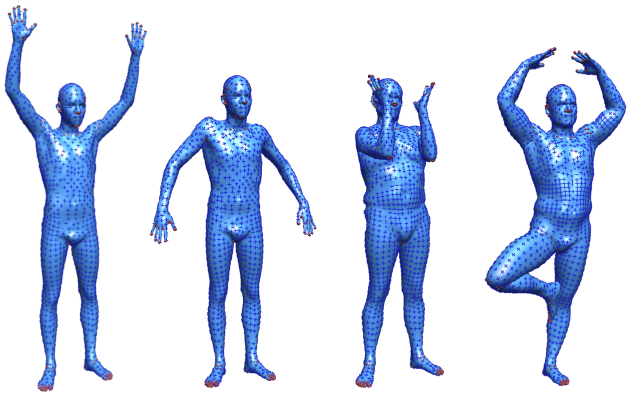


Figure 8. The smooth frame fields (shown as crosses) aligned with salient geometric features have strong consistency among diverse human body shapes. The singular vertices marked as red points are also distributed similarly across the shapes, concentrating on regions of high curvature, e.g. nose, finger tips, and toes.

Table 7. Testing accuracy of different frame alignment choices, on the FAUST non-rigid registration task by classification. SF means smoothness only without alignment to surface anisotropy. The other numbers are used as the curvature direction alignment weight $\lambda$ for computing the smooth frame field.

| | SF | 0.01 | 0.1 | 1 |
|---|---|---|---|---|
| Accu.(%) | 88.56 | 92.01 | 91.97 | 90.77 |

## C. Tangent plane projection and resampling

The algorithm for building the convolution structure on a local patch of a mesh vertex is illustrated in Alg. 1 in pseudo code. For each mesh vertex, the algorithm first does a flood searching of $K$ neighbor vertices in $O(K)$ and projects the vertices onto the tangent plane using local coordinate systems. It then triangulates the projected vertices into a Delaunay triangulation in $O(K \log K)$, and samples $H \times W$ grid points against the triangulation in $O(HW \log K)$. The sampled grid points are finally stored into the sparse tensor that will be reshaped as a sparse matrix and readily multiplied with feature maps in each convolution operation (Sec. 5.2). Note that all vertices can be processed in parallel.

In the algorithm we have abused notations slightly, using $t[0]$ of a tuple to represent the vertex, its index, and its spatial position; the exact meaning should be clear from context. For a given level of domain resolution, the patch size parameter $d$ is set to be the average edge length of all meshes in the given level of the training dataset.

**Algorithm 1:** Tangent plane projection and feature map resampling for a local patch

**Input:** $v_i \in V$, frames $F$, transport $\tau$, patch side length $d$, conv kernel shape $H \times W$
**Output:** updated sparse tensor $S$ of shape $|V| \times N \times H \times W \times |V| \times N$

```
// Flood to find and project neighbor vertices
```
$Q = [(v_i, (0,0), 0)], P = \{\}, visited = \{v_i\}$;
**while** $Q$ *not empty* **do**
  $t$ =dequeue($Q$), $P = P \cup t$;
  **if** $dist(v_i, t[0]) > \sqrt{2}d$ *or* $\|t[1]\| > \sqrt{2}d$ **then**
    continue;
  **end**
  **for** $v_k \sim t[0], v_k \notin visited$ **do**
    $visited = visited \cup v_k$;
    $\mathbf{u}^l_{v_k} = \tau_{t[0],v_k}(\mathbf{u}^{t[2]}_{t[0]})$;
    $\mathbf{v}'_k = 0.5 \cdot (F^{t[2]}_{t[0]} + F^l_{v_k})(\mathbf{v}_k - t[0]) + t[1]$;
    enqueue($Q$, $(v_k, \mathbf{v}'_k, l)$);
  **end**
**end**

```
// Triangulate the projected points
```
$DT(P)$ = Delaunay triangulation of $\{t[1] | t \in P\}$;

```
// Resample with a regular grid sized d×d
```
**for** $j = 1, \cdots, N$ **do**
  **for** grid point $p_{r,c}$, $1 \le r \le H, 1 \le c \le W$ **do**
    find the containing triangle in $DT(P)$ with vertices corresponding to $(t_a, t_b, t_c) \subset P$;
    compute barycentric weights $(w_a, w_b, w_c)$;
    $S(i, j, c, r, t_a[0], t_a[2]) = w_a$;
    $S(i, j, c, r, t_b[0], t_b[2]) = w_b$;
    $S(i, j, c, r, t_c[0], t_c[2]) = w_c$;
  **end**
**end**

## D. Network structures and training details

We have used convolution kernels with spatial size $5 \times 5$ for all deformable domain tasks, and $3 \times 3$ for the semantic scene segmentation. All our networks have been trained with the Adam solver [20] and batch size one, with fixed learning rate $10^{-4}$.

The network structure used for SHREC'15 non-rigid shape classification is shown in Fig. 9. It is trained for 50 epochs on a single GPU. For the variant network without any normalization layers, it needs to train for 100 epochs until convergence.

The network used for the human body segmentation task is shown in Fig 10. It has three level-of-details. The loss function is the summation of cross entropy between predicted segmentation label and the ground truth label for each mesh vertex. The network is trained for 50 epochs. To obtain the predicted per-face segmentation labels, we sample points for each face of a test mesh and project the points onto closest vertices of our remeshed models, whose labels are used to vote for the face label of the original test mesh.

The network used for the human body registration task by vertex classification, and testing different frame field symmetry ordersis is shown in Fig. 11. The network is trained for 400 epochs.

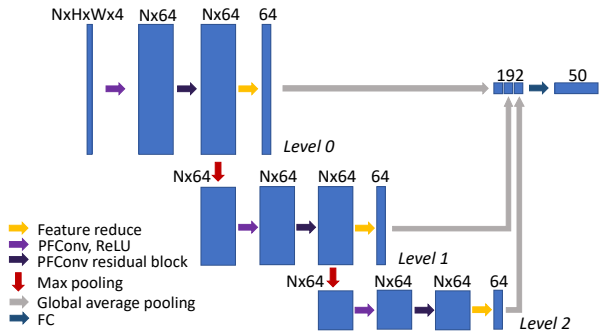



Figure 9. The network used for SHREC'15 non-rigid shape classification task. Each box represents a feature map of shape $V \times C$, where $C$ is the total feature size for all $N$=4 cover sheets and given by numbers aside the boxes, and $V$ the number of surface vertices. The input feature map is a 4-channel feature of $H \times W$ grid points for each vertex (Sec. 5.2). The "convolution through residual block" contains two sequential residual blocks, with each block made by two convolutions that retain the input feature size. All convolution operations except the last one are followed with instance normalization and ReLU. Global average pooling is a standard average pooling over all vertices. For this dataset there are around 10k, 1700, 300 vertices for the three level-of-details, respectively.

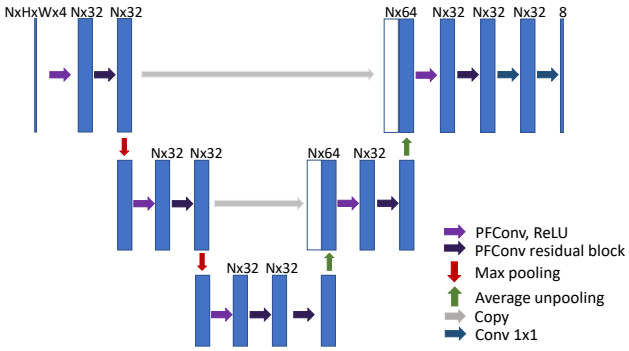



Figure 10. The network used for the human body segmentation task. See caption of Fig. 9 for detailed explanation. The number of vertices for the three level-of-details are $V, V/3, V/9$, where $V$ is the number of vertices of each mesh in the dataset. In the original dataset, $V$ varies from 3k to 12k. For the remeshed data, $V$ is around 7k.

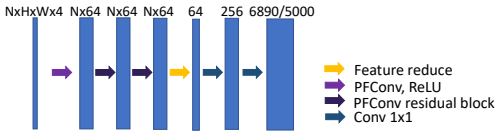



Figure 11. The network used for human body registration through a classification of mesh vertices into 6890 or 5000. The number of surface vertices is 6890 for the original dataset and 5000 for the remeshed dataset. See caption of Fig. 9 for detailed explanation.

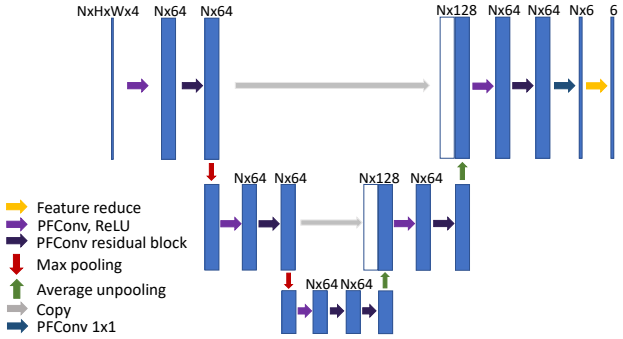



Figure 12. The regression network used for human body regression task. See caption of Fig. 9 for detailed explanation. The number of surface vertices are around 10k, 3.2k, 1k for the three level-of-details respectively.

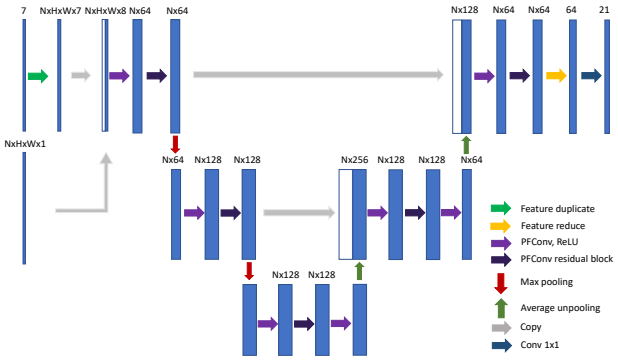



Figure 13. The network used for ScanNet segmentation task. See caption of Fig. 9 for detailed explanation. The input include the 7-channel feature of each vertex and the 1-channel local height feature for each grid point. The number of vertices for the three level-of-details are $V, V/3, V/9$, where $V$ is the number of vertices of each cropped chunk, with the crop method same as [16].

The network used for the ScanNet semantic scene segmentation task is shown in Fig. 13. The network outputs, for each vertex, the probability distribution of 21 segmentation labels, which is compared with ground truth label using cross entropy during training. It is trained for 30 epochs.

## E. More results and comparisons

**Shrec'15 classification** We show some results in the classification task in Fig. 14; the single misclassified shape by our method is a challenging "ant" that looks similar to the wrong label "spider".

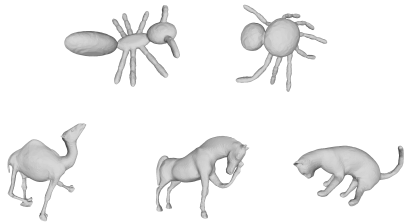


Figure 14. In the first row left shows the single incorrectly classified shape by our method; it is an "ant" misclassified as "spider" (an example shown on the right, is indeed confusingly similar to "ant"). In the second row we show more shapes in the SHREC'15 dataset, which are "camel", "horse" and "cat".

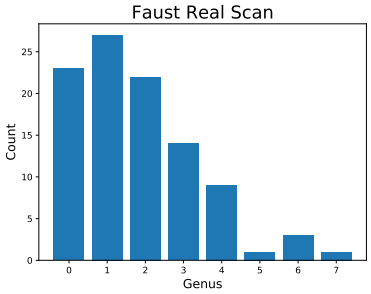



Figure 15. Genus of the meshes in the FAUST real scan dataset. More than half of the meshes have genus larger than 1.

**Non-rigid registration by fitting template embedding.** In this part we present an application that resolves the non-rigid registration problem with an approach different from the per-vertex classification (Sec. 6.1). We notice that the registration by classification has severe limitations in real applications: to classify each vertex to 6k classes for example is not scalable when there are many input vertices, and the classification error does not measure at all how far away a mis-classified vertex is from ground-truth. Thus we propose a novel but simple method for non-rigid registration that uses a surface-based CNN for direct regression of the template embedding in $\mathbf{R}^3$.

We evaluate on the real scans of the FAUST dataset, which has 80 meshes for training and 20 for test. The meshes are noisy, with diverse and high genus for different poses of a same person (see Fig. 15 for statistics), which is frequently due to the merging of spatially intersecting components. Since the raw scans are very dense meshes, we have remeshed each raw scan to simpler meshes with the number of vertices around 10k. For each real scan there is a registered deformed template mesh, which provides the ground truth embedding for supervision and testing. To be specific, we project a vertex of the real scan to the closest point on the registered deformed template mesh, and take

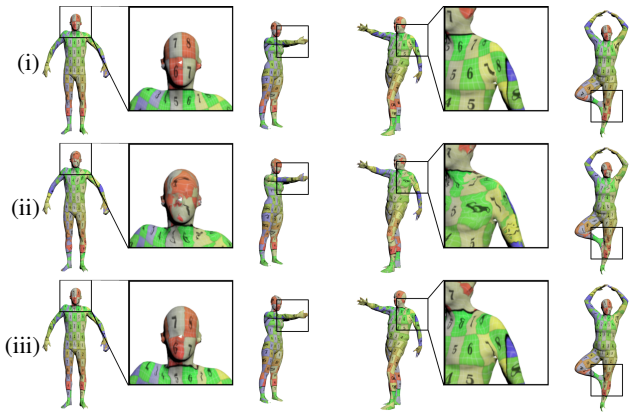



Figure 16. Results of non-rigid human body registration through regression of the template embedding coordinates. (i) shows the texture mapping using the groundtruth correspondence. (ii) is the results of [32]. (iii) shows our results.

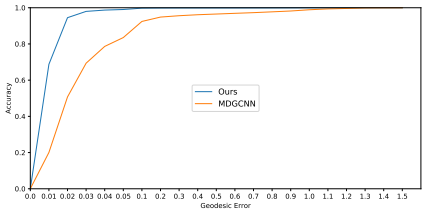



Figure 17. The ratio of vertices whose error is bellow given threshold. The per-vertex error is the geodesic distance between the predicted point position and ground truth on the template surface, normalized by square root of surface area. Our accuracy under 0.03 is 97.98% while [32] is 69.37%.

its position and normal vectors on the rest pose template as the supervising regression target.

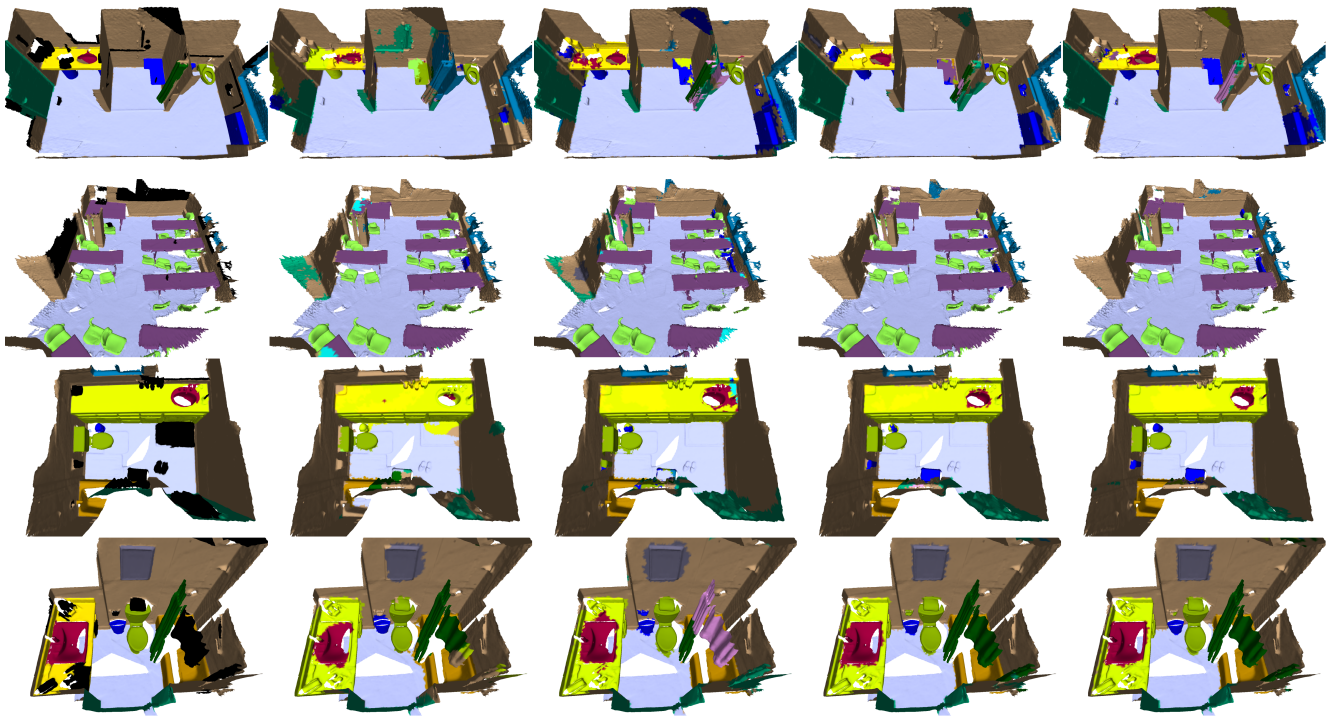
The network for this point-wise regression is a standard UNet structure as shown in Fig. 12. For each vertex of an input raw scan mesh, the output contains the position and normal vectors of the corresponding point on the rest pose template mesh. The training loss is

$$
\begin{aligned}
L = & \frac{1}{V}\sum_{i}^{V}\left(\|\mathbf{p}_i - \mathbf{p}_i^0\|_1 + \frac{w_{reg}}{V_i}\sum_{j\sim i}\|\mathbf{p}_i - \mathbf{p}_j\|_1\right) \\
& + \frac{w_n}{V}\sum_{i}^{V}\left(\|\mathbf{n}_i - \mathbf{n}_i^0\|_1 + \frac{w_{reg}}{V_i}\sum_{j\sim i}\|\mathbf{n}_i - \mathbf{n}_j\|_1\right) \\
& + \frac{w_{con}}{E}\sum_{i\sim j}|\mathbf{n}_i \cdot (\mathbf{p}_i - \mathbf{p}_j)|,
\end{aligned}
$$

where $V$ is the number of vertices of the raw scan mesh, $\mathbf{p}$ the regressed vertex position, $\mathbf{p}^0$ the target position, $\mathbf{n}$ the regressed vertex normal, $\mathbf{n}^0$ the target normal, $w_n = 0.1$ to normalize different scales between position and normal in the dataset, $w_{reg} = 0.2$ the weight for Laplacian regularization terms of position and normal, $w_{con} = 20$ the weight for normal and position consistency, $V_i$ the number of neighboring vertices of the $i$-th vertex, and $E$ the number of directed mesh edges. We use $l_1$ norm for these losses because there are noisy vertices in the raw scans which do not have valid target points on the template surface. We train the network for 200 epochs on single GPU using Adam solver with a fixed learning $1 \times 10^{-4}$.

Geodesic errors of the network predictions on the test set are shown in Fig. 17. Following [19], the geodesic error for a surface point $x$ with predicted position $y$ and ground truth point $y^*$ on the template surface $\mathcal{M}$ is computed as $\epsilon(x) = \frac{d_{\mathcal{M}}(y,y^*)}{\sqrt{|\mathcal{M}|}}$, where $d_{\mathcal{M}}(\cdot,\cdot)$ computes the geodesic distance of two points projected onto the surface $\mathcal{M}$, and $|\mathcal{M}|$ is its area for normalization. Visual results are shown in Fig. 16. It is clear that our results are better than MDGCNN both quantitatively and qualitatively on these real scans, and the difference seems to be more obvious than the registration by vertex classification task on the clean meshes (Sec. 6.1).

**More results of ScanNet segmentation.** We present the per-category prediction accuracy (measured by IoU) of comparing methods for ScanNet semantic segmentation in Table 8 and Table 9. For Ours*, the network structure is similar to the network shown in Fig. 13, but each "PFConv residual block" contains three sequential residual blocks and the feature sizes in three levels are changed to 128, 256, 512, respectively. More visual results are shown in Fig. 18.



(i) (ii) (iii) (iv) (v)

Figure 18. More results of Scannet segmentation.(i) is the ground truth segmentation; (ii) is the results of [42] (iii) is the results of [16]; (iv) shows our results. (v) is the result of our method with deeper network. Our method gives clearer boundaries, like the boundary between window and wall in the second row, the boundary between picture and wall and the boundary of sink in the last row.

Table 8. Per-category IoU on ScanNet validation set. The abbreviations respectively stand for "bathtub, bed, bookshelf, cabinet, chair, counter, curtain, desk, door, floor, otherfurniture, picture, refrigerator, shower, curtain, sink, sofa, table, toilet, wall, window". The highest accuracies both among the three comparing results and among the four comparing results with our additional increased network are marked in bold.

| Method | mIoU | bath | bed | book | cab | chr | cntr | crtn | desk | door | flr | other | pic | refrg | shwr | sink | sofa | tab | toil | wall | wdw |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [42] | 49.1 | 68.0 | 63.8 | 56.3 | 41.7 | 73.6 | 45.6 | 33.2 | 40.7 | 34.9 | 91.9 | 26.2 | 14.5 | 31.7 | 28.1 | 44.2 | 62.8 | 51.5 | 68.8 | 67.9 | 38.3 |
| [16] | 58.1 | 67.6 | 67.3 | 71.3 | 46.8 | 78.1 | 44.4 | 52.5 | 47.5 | 44.8 | **94.4** | 40.2 | 21.1 | 35.2 | 51.3 | 51.7 | 64.0 | **63.5** | 80.3 | 75.0 | 46.0 |
| Ours | **63.3** | **79.7** | **70.3** | **73.7** | **55.6** | **81.0** | **53.9** | **70.1** | **53.1** | **50.0** | 93.7 | **42.3** | **30.3** | **46.3** | **55.6** | **60.1** | **66.4** | 60.9 | **87.3** | **78.7** | **56.5** |
| Ours* | **66.2** | **81.6** | **73.0** | **77.0** | **56.8** | **83.3** | **62.8** | **70.9** | **55.8** | **52.3** | 94.0 | **46.4** | **33.1** | **51.7** | **60.9** | **61.2** | **72.3** | **65.0** | **87.7** | **80.0** | **58.6** |

Table 9. Per-category IoU on ScanNet test set. See caption of Table 8 for explanations.

| Method | mIoU | bath | bed | book | cab | chr | cntr | crtn | desk | door | flr | other | pic | refrg | shwr | sink | sofa | tab | toil | wall | wdw |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [42] | 43.8 | 43.7 | 64.6 | 47.4 | 36.9 | 64.5 | 35.3 | 25.8 | 28.2 | 27.9 | 91.8 | 29.8 | 14.7 | 28.3 | 29.4 | 48.7 | 56.2 | 42.7 | 61.9 | 63.3 | 35.2 |
| [16] | 56.6 | 67.2 | 66.4 | 67.1 | 49.4 | 71.9 | **44.5** | 67.8 | 41.1 | 39.6 | 93.5 | 35.6 | 22.5 | 41.2 | **53.5** | 56.5 | 63.6 | 46.4 | 79.4 | 68.0 | 56.8 |
| Ours | **60.2** | **74.6** | **71.2** | **67.4** | **53.5** | **75.6** | 41.6 | **68.1** | **42.0** | **43.4** | **93.8** | **40.1** | **27.0** | **51.2** | 51.1 | **61.2** | **69.4** | **48.3** | **84.7** | **77.7** | **61.5** |
| Ours* | **62.2** | **79.7** | 69.7 | **75.0** | **57.7** | **79.2** | **47.6** | **68.5** | 36.6 | **46.8** | **94.2** | **41.4** | **30.7** | **53.2** | 49.4 | **68.1** | **71.5** | 47.5 | **88.0** | **79.6** | 59.3 |