EFEM: *E*quivariant Neural *F*ield *E*xpectation *M*aximization for 3D Object Segmentation Without Scene Supervision — Supplementary Material —

Jiahui Lei[1] Congyue Deng[2] Karl Schmeckpeper[1] Leonidas Guibas[2] Kostas Daniilidis[1]
[1] University of Pennsylvania [2] Stanford University
{leijh, karls, kostas}@cis.upenn.edu, {congyue, guibas}@cs.stanford.edu

This supplementary aims to provide more details of our method (Sec. S.1,S.2), more information on our dataset (Sec. S.4), and more visual results (Sec. S.5) and complete tables (Sec. S.6). We also provide more ablation studies and analysis in Sec. S.3

S.1. Details on the Equivariant Shape Prior

Here we explain the architectural design and implementation details of our equivariant shape prior. In Section S.1.1, we introduce the equivariant point cloud encoder, which takes a point cloud $P_{N_O\times 3}$ as input and outputs the global equivariant latent code Θ. Then in Section S.1.2, we will introduce the implicit decoder which decodes Θ together with any $x_{1\times 3}$ 3D query point to get its SDF value.

S.1.1. Encoder Networks

Before being fed into the encoder, the input point cloud $P_{N_O\times 3}$ first has its centroid $\bar{P}$ subtracted from it for translation equivariance. This centers the point cloud at the origin.

In Section S.1.1.1, we will explain the construction of the backbone building blocks. In Section S.1.1.2, we will show how to compute the final latent embedding outputs from the backbone output.

S.1.1.1 Point Cloud Backbone

The backbone of our encoder is constructed from a stack of equivariant blocks with each block aggregating information from the point neighborhoods. We denoted the Mth block as $L^{(M)}$. The input to each block is a set of vector channel features (for each point) of shape $C_{\text{in}} \times 3$, denoted as $\mathcal{F}^{(M)} = \{F_i^{(M)}\}$, $i = 1, 2, \dots N$ where N is the number of points in the point cloud. Similarly, the output is a set of new vector channel features $\mathcal{F}^{(M+1)} = \{F_i^{(M+1)}\}$ with shape $C_{\text{out}} \times 3$. Each block first fuses local information in the point neighborhood and then fuses global information with global pooling. We first explain two design choices of the local fusion: VN-DGCNN aggregation and VN-Transformer aggregation, and then provide details of the global fusion. When entering the block, we first find the K-nearest neighbors for each point in the feature space as the VN-DGCNN [7, 10].

VN-DGCNN aggregation [7] We use this aggregation for all the tabletop (mug and kitchen container) object shape priors in our main paper experiments. The aggregation is similar to [7]. We compute edge features in the KNN neighborhood and use a symmetric operation to fuse all the edges:

$$F_i^{(M+1)} = \frac{1}{|\mathcal{N}_i|} \sum_{(i,j)\in\mathcal{N}_i} \text{VNLA}\left((F_j^{(M)} - F_i^{(M)}) \oplus F_i^{(M)}\right). \tag{1}$$

Here $\mathcal{N}_i$ is the KNN neighborhood of the ith point, and $\oplus$ means concatenation of the feature. VNLA() means a standard Vector Neuron Linear layer followed by a standard Vector Neuron Activation layer. In our implementation, we use VN-LeakyReLU activations and never use BatchNorm.

VN-Transformer aggregation [1] We use this aggregation mode for the chair shape prior in our main paper. To compute the new output feature for point i, we first compute Q, K, V. For point i:

$$Q_i = \text{VNLA}(F_i^{(M)}), \tag{2}$$

and for all i's k-nearest neighbors:

$$K_j = \text{VNLA}((F_j^{(M)} - F_i^{(M)}) \oplus F_i^{(M)}) \tag{3}$$

$$V_j = \text{VNLA}((F_j^{(M)} - F_i^{(M)}) \oplus F_i^{(M)}). \tag{4}$$

Here we also input the information about the query $F_i^{(M)}$ to the computation of K and V since we want to maximize the expressivity. This is not a critical design choice as other variants of transformer or aggregation should also work. Next, we discuss the invariant transformer attention weight and the scale invariance.

Figure S1. Our shape Encoder-Decoder is equivariant to SIM(3) transformations. When the input point clouds (Yellow) are scaled, rotated, and translated, our output shapes (Blue) are also correspondingly scaled, rotated, and translated.

The scale invariance can be easily obtained by following the procedure from [3]. Note that our network does not have bias terms, and all VN Linear maps maintain the scale equivariance. Since we use VN-LeakyReLU (or VN-ReLU), it is clear from [3] that such activation also maintains scale equivariance. However, when we want to take the inner product of two rotation-scale equivariant features to form the invariant attention weight [1], the inner product will count the scale twice and is therefore not invariant. [3] proposes a channel-wise normalization to get scale invariant but rotation equivariant vector channel features in Section 5 of [3]. Given a scale and rotation equivariant vector channel feature $F_{C\times 3}$, we first compute the direction $\hat{F}_{C\times 3}$ and length $\widetilde{F}_{C\times 1}$ of each vector channel in F. Then we factor out the scale by regarding $\widetilde{F}_{C\times 1}$ as a 1D vector and normalizing across the channel dimension C, so the output scale invariant but rotation equivariant feature is:

$$F' = \text{ChNorm}(F) = \hat{F}_{C\times 3} \odot \frac{\widetilde{F}_{C\times 1}}{\|\widetilde{F}_{C\times 1}\|}, \tag{5}$$

where $\odot$ means the per channel multiplication. We use ChNorm() to exclude the scale factor from K_j and Q_i so the attention weight for point i is neighbor j:

$$w_j = \text{Softmax}_j\left(\frac{1}{\sqrt{3C}}\langle \text{ChNorm}(Q_i), \text{ChNorm}(K_j)\rangle\right), \tag{6}$$

where $\langle , \rangle$ is the full inner product (not channel-wise, including summation over channel dimension), which guarantees the weight is both rotation and scale invariant. The final output feature for ith point is:

$$F_i^{(M+1)} = \sum_{(i,j)\in\mathcal{N}_i} w_j V_j, \tag{7}$$

which is rotation and scale equivariant. We also use multi-head attention by dividing the overall channel dimensions into several groups and computing the attention weight per channel group.

Global context [7] While each block can propagate and aggregate information in the KNN neighborhood, we also want to consider the global context to enhance the expressivity of our encoder. After getting $\mathcal{F}^{(M+1)}$, we perform a global average pooling over all the point features and get the averaged global feature $\bar{F}^{(M+1)}$, so the final output features for each block is:

$$F'^{(M+1)} = \text{VNLA}(F^{(M+1)} \oplus \bar{F}^{(M+1)}), \tag{8}$$

where $\oplus$ means channel concatenation.

For efficiency, we also use Farthest Point Sampling [8], to downsample the point cloud between layers leading to a coarse-to-fine encoding. The encoder used in the experiments has 7 blocks and the output feature dimension of each block is $[32, 32, 64, 64, 128, 256, 512]$.

After the final block, we apply average pooling over all point feature outputs from the last block and use the result as the output global encoding of our equivariant backbone, which is denoted as F in our main paper Sec.3.1. Note that F is vector channeled (shape $C\times 3$) and is rotation and scale equivariant, but translation invariant, which means for any rotation R, translation t and scale s, if the un-transformed point cloud P has been encoded as F then the transformed point cloud $sPR + t$ is guaranteed to be encoded as sFR.

S.1.1.2 Final Outputs

As explained in Sec.3 in the main paper, the output of the encoder includes several latent embeddings Θ computed with separated small network branches at the end of the backbone. Denote the backbone output as F, the latent embeddings are defined as follows:

1. The rotation equivariant global latent code Θ_{R}:

$$\Theta_{\text{R}} = \text{ChNorm}\left(\text{VNL}(F)\right), \tag{9}$$

where VNL denotes a standard VN Linear layer and Θ_{R} has shape 256×3 in our implementation. As proven by [1, 3, 7], if the input point cloud is transformed by any SIM(3) element (s, R, t), then the new transformed rotation latent code is:

$$\begin{aligned}\Theta'_{\text{R}} &= \text{ChNorm}\left(\text{VNL}(sFR)\right) \\ &= \text{ChNorm}\left(\text{VNL}(F)\right) R = \Theta_{\text{R}} R\end{aligned} \tag{10}$$

2. The invariant global latent code Θ_{inv}:

$$\Theta_{\text{inv}} = \langle \text{ChNorm}(\text{VNL}(F)), \Theta_{\text{R}} \rangle_{\text{Channel}}, \tag{11}$$

where $\langle , \rangle_{\text{Channel}}$ denotes the channel-wise inner product. The resulting invariant latent code has a scalar per channel and has shape 256×1 in our implementation. Similarly, if the input is transformed by (s, R, t), the new invariant code stays the same:

$$\begin{aligned}\Theta'_{\text{inv}} &= \langle \text{ChNorm}(\text{VNL}(sFR)), \Theta_{\text{R}} R \rangle_{\text{Channel}} \\ &= \langle \text{ChNorm}(\text{VNL}(F))R, \Theta_{\text{R}} R \rangle_{\text{Channel}} \\ &= \langle \text{ChNorm}(\text{VNL}(F)), \Theta_{\text{R}} \rangle_{\text{Channel}} = \Theta_{\text{inv}}\end{aligned} \tag{12}$$

3. The explicit scale scalar Θ_{s}: This output is for querying the SDF because the query position should first eliminate the scale and then be sent to the MLP decoder. Θ_{s} is a single scalar output calculated by first finding the per-channel vector length $\widetilde{F}_{C\times 1}$ and then averaging over the channel:

$$\Theta_{\text{s}} = \frac{\sum_c \widetilde{F}_c}{C}. \tag{13}$$

When the input is transformed by (s, R, t), the new scale is:

$$\Theta'_{\text{s}} = \frac{\sum_c \widetilde{(sFR)}_c}{C} = s\frac{\sum_c \widetilde{F}_c}{C} = s\Theta_{\text{s}} \tag{14}$$

4. The corrected centroid Θ_{c}: although the translation equivariance is naturally obtained by subtracting the centroid, during decoding, the query position for the SDF must also subtract the centroid to eliminate the translation factor. However, when the observation is partial, noisy, or unbalanced, the centroid will shift by a large amount, so we let the network predict a centroid correction vector to stabilize the shape prior:

$$\Theta_{\text{c}} = \text{VNL}(F) + \bar{P}, \tag{15}$$

where the VN linear layer here has only one channel output. Θ_{c} has shape 1×3. Note that even without any training, the predicted corrected centroid guarantees the full SIM(3) equivariance, i.e. when the input point cloud is transformed by a SIM(3) element (s, R, t), the newly predicted center is:

$$\begin{aligned}\Theta'_{\text{c}} &= \text{VNL}(sFR) + (s\bar{P}R + t) \\ &= s\text{VNL}(F)R + s\bar{P}R + t \\ &= s(\text{VNL}(F) + \bar{P})R + t = s\Theta_{\text{c}}R + t\end{aligned} \tag{16}$$

In summary, the output of the encoder is:

$$\Theta = (\Theta_{\text{R}}, \Theta_{\text{inv}}, \Theta_{\text{c}}, \Theta_{\text{s}}) = \Phi(P), \tag{17}$$

which contains all the information for a SIM(3)-transformed shape. For any transformation $g \in SIM(3)$ represented as scale, rotation and translation (s, R, t) the encoder strictly guarantees the equivariance and the shape embeddings Θ obey transformations:

$$\Theta' = g \circ \Theta = (\Theta_{\text{R}}R, \Theta_{\text{inv}}, s\Theta_{\text{c}}R + t, s\Theta_{\text{s}}) = \Phi(sPR + t). \tag{18}$$

S.1.2. Decoder Networks

Given a query position $x_{1\times 3}$ in the 3D space, we first canonicalize it to an invariant embedding along with the shape codes and then process it with a standard MLP.

$$\begin{aligned}\widehat{SDF}(x; \Phi(P)) &= \widehat{SDF}(x; \Theta) \\ &= \text{MLP}\left(\left\langle \Theta_{\text{R}}, \frac{x - \Theta_{\text{c}}}{\Theta_{\text{s}}} \right\rangle_{\text{channel}} \oplus \Theta_{\text{inv}}\right),\end{aligned} \tag{19}$$

where the P denotes the input point cloud, the MLP denotes a standard non-equivariant MLP, and $\langle , \rangle_{\text{channel}}$ means the channel-wise inner product. $\oplus$ here means the channel concatenation.

To show that the SDF is equivariant, we show that given any SIM(3) element denoted as scale, rotation, and translation $g = (s, R, t)$ the model strictly has:

$$\widehat{SDF}(x; \Phi(P)) = \widehat{SDF}(sxR + t; \Phi(sPR + t)), \tag{20}$$

since we showed that the encoder is equivariant in the

Section S.1.1, we can show:

$$\begin{aligned}
&\widehat{SDF}(sxR+t;\Phi(sPR+t)) = \widehat{SDF}(sxR+t;g\circ\Theta)\\
&= \text{MLP}\left(\left\langle \Theta_{\text{R}}R, \frac{sxR+t-(s\Theta_{\text{c}}R+t)}{s\Theta_{\text{s}}}\right\rangle_{\text{channel}} \oplus \Theta_{\text{inv}}\right)\\
&= \text{MLP}\left(\left\langle \Theta_{\text{R}}R, \frac{(x-\Theta_{\text{c}})R}{\Theta_{\text{s}}}\right\rangle_{\text{channel}} \oplus \Theta_{\text{inv}}\right)\\
&= \text{MLP}\left(\left\langle \Theta_{\text{R}}, \frac{x-\Theta_{\text{c}}}{\Theta_{\text{s}}}\right\rangle_{\text{channel}} \oplus \Theta_{\text{inv}}\right) = \widehat{SDF}(x;\Phi(P)).
\end{aligned} \tag{21}$$

Therefore, our shape prior is equivariant to SIM(3) transformation, as shown in Fig. S1.

S.1.3. Shape Prior Training Details

SDF Loss The main loss is the average of the SDF L1 regression loss on each query point $\mathcal{L}_1(x) = |\widehat{\text{SDF}}(x) - \text{SDF}^*(x)|$ where $\text{SDF}^*(x)$ is the ground truth SDF values. Similar to DISN [13], we weight the near-surface regression losses more to encourage near-surface accuracy:

$$\mathcal{L}_{\text{SDF}} = \frac{\lambda_{\text{near}}\sum_{x\in\mathcal{Q}_{\text{near}}}\mathcal{L}_1(x) + \lambda_{\text{far}}\sum_{x\in\mathcal{Q}_{\text{far}}}\mathcal{L}_1(x)}{|\mathcal{Q}_{\text{near}}| + |\mathcal{Q}_{\text{far}}|}. \tag{22}$$

Here $\mathcal{Q}_{\text{near}}$ is the set of query points whose absolute ground truth SDF is smaller than a threshold (0.1 in our experiments) and $\mathcal{Q}_{\text{far}}$ is the rest of the query points. The balance weights in our experiments are $\lambda_{\text{near}} = 1.0$ and $\lambda_{\text{far}} = 0.5$.

Regularization Loss Our network guarantees scale and translation equivariance without any training. However, Θ_{c} and Θ_{s} are required to factor out the translation and scale from each query position. We observe that letting the network freely predict them will result in unstable optimization in the early epochs of the shape prior training. As a result, we have to regularize them. ShapeNet is a canonicalized dataset where all the meshes are aligned so we can regularize all the predicted Θ_{s} to be 1.0:

$$\mathcal{L}_{\text{scale}} = |1.0 - \Theta_{\text{s}}|. \tag{23}$$

Additionally, we have to apply a loss on Θ_{c} to correct the centroid. As mentioned above, when the input is partial the centroid is not a good reference origin for the shape SDF. Since all the meshes are aligned in ShapeNet, we can directly regularize the predicted center to lie at $[0.0, 0.0, 0.0]$:

$$\mathcal{L}_{\text{center}} = \|\Theta_{\text{c}}\|_2. \tag{24}$$

When the input point cloud is partial, this loss enforces the network to correct the center from the centroid to the object center. Even without the above loss, Θ_{c} also strictly guarantees the translation equivariance, but may not point to the object center.

In summary, the total loss is:

$$\mathcal{L} = \omega_{\text{SDF}}\mathcal{L}_{\text{SDF}} + \omega_{\text{center}}\mathcal{L}_{\text{center}} + \omega_{\text{scale}}\mathcal{L}_{\text{scale}}, \tag{25}$$

and in our experiments, we set $\omega_{\text{SDF}} = 1.0$, $\omega_{\text{center}} = 0.2$ and $\omega_{\text{scale}} = 0.001$.

Augmentation Although our network strictly guarantees SIM(3) equivariance, the content of real-world observations are always partial and noisy. Therefore we apply several, mainly content-wise, augmentations for the input point cloud to let our shape prior better generalize to the real world:

1. **Depth point cloud**: We use a synthetic camera to randomly render depth images from different viewpoints. The input point cloud is sampled from several depth back-projections.

2. **Random crop addition**: We randomly add small ball crops of other meshes in the training set to the object's neighborhood to make the network more robust to crowded data.

3. **Random crop removal**: We randomly remove some ball crops on the object surface.

4. **Random plane addition**: For the chair shape-prior training, we also randomly add some small horizontal or vertical planes to the neighborhood of the object.

5. **Centroid augmentation**: To robustify the Θ_{c} prediction, we simulate a larger centroid shift by adding a small gaussian noise to the computed input point cloud centroid $\bar{P}$.

6. **Other noise**: We also add small gaussian noise to every point.

S.2. Details on EFEM Implementation

S.2.1. Shape and pose output

As mentioned in Sec.3.2 in the main paper, our model also provides mesh reconstruction and pose estimation for each detected object. The implicit shape outputs with latent Θ of the last iteration are converted to triangle meshes via marching cube, which is visualized as colored meshes in the figures in the main paper.

Since our network is fully equivariant, to output the absolute pose of each object, we should have the knowledge of what is the "absolute object coordinate frame". Note that all the meshes in the training set (ShapeNet [2]) are aligned, so we exploit the alignment of the training set to output the absolute pose estimation. Once the network is trained, for every shape point cloud input P_j in the training set $\mathcal{X}_{\text{train}}$,

we use our encoder Φ to encode all the shapes and form a codebook:

$$\text{CodeBook} = \{\Theta_j = \Phi(P_j) | P_j \in \mathcal{X}_{\text{train}}\}. \quad (26)$$

Note that the codebook can be directly computed once the shape prior is trained and saved to the file cache. Given the proposal's last Θ estimation, we first find the K-nearest-neighbors (e.g. $K = 16$ in our mugs experiments) in the latent space of Θ_{inv}. Then we align the point cloud of each of the training set neighbors to our currently estimated shape. Following [15], we solve a simple Procruste problem on Θ_{R} by treating each channel as a correspondence pair. We denote the kth neighbor's latent code as $\Theta^{(k)}$ and find an $O(3)$ transformation (including flipping):

$$\mathbf{U}, \mathbf{S}, \mathbf{V}^T = \text{SVD}\left(\sum_c \Theta_{\text{R}}[c]^T \Theta_{\text{R}}^{(k)}[c]\right), \quad (27)$$

where $\Theta_{\text{R}}[c]$ with shape 1×3 is the vector at channel c, then the $O(3)$ transformation prediction is:

$$\widehat{R} = \mathbf{U}\mathbf{V}^T. \quad (28)$$

To align the training set nearest neighbor shape point cloud P_k from the ShapeNet canonical frame to our current estimated shape, we apply:

$$P'_k = \frac{\Theta_{\text{s}}}{\Theta_{\text{s}}^{(k)}}(P_k - \Theta_{\text{c}}^{(k)})\widehat{R}^T + \Theta_{\text{c}}. \quad (29)$$

After the latent registration, we also use several ICP steps between the training shape point cloud and our estimated shape point cloud to explicitly refine the rotation. We select the most similar shape from the K neighbors by selecting the one with the smallest Chamfer Distance to our estimated shape. We use the transformation between our estimated shape and the selected best-matched shape in the training set as an approximate pose and size estimation for the detected object. Given the mesh, an oriented bounding box can also be computed from the pose and the shape. The poses and bounding boxes are shown in all figures in our main paper.

S.2.2. Proposal filtering

Initialization and Steps To ensure that all objects of interest are covered by some proposal initialization, we initialize far more proposals than the number of objects that might appear in the scene. For example, we start with 400 initial random ball crops for the mug experiments. Fortunately, many of these proposals can be eliminated after a small number of steps. Although we observe that the iterations converge quickly, we set the Phase-1 independent steps to run 15 iterations and Phase-2 joint steps to run 5 iterations to ensure full convergence.

Unreasonable elimination : At every EM iteration, we count the number of points in the proposal that have smaller distance errors and normal errors than the corresponding thresholds (e.g. 0.1 and 60° for mugs experiments) and eliminate the proposals with too few of these inlier points (e.g. 300 for mugs experiments). Another straightforward check we apply is on the shape size. Since it is not easy to check the shape size directly in the SDF, we use the above-mentioned bounding box computed from the shape and pose by-products to eliminate unreasonable shape proposals. For example, the real-mug experiments have the reasonable $X - Y$ radius inside $[0.03m, 0.08m]$ and the reasonable Z radius inside $[0.015m, 0.1m]$. The shape size elimination happens at the end of Phase 1 and the end of Phase 2.

Duplication and inclusion elimination : As mentioned in Sec.3.3 of our main paper, we detect and resolve duplications between multiple proposals, which significantly reduces the number of proposals we have to maintain after several early steps. Given a proposal, we identify a binary inlier mask over all the scene points based on the decoded distance error (Eq.4 in the main paper). If $e_D(\mathbf{x}; \Theta) < \delta_{\text{dup}}$ we mark the point as an inlier of the current proposal. δ_{dup} is the inlier threshold and we set it to 0.04 in the mugs experiments. We compare each proposal with all other proposals, if the IoU between two binary inlier masks is larger than a threshold (e.g. 70% for mugs experiments), we identify them as duplicated and eliminate the one with a lower current fitting score (Eq.8 in the main paper).

Similarly, during each iteration in Phase 2, we traverse all the proposals and if more than $\delta_{\text{inclusion}}$ (e.g. 80% in mugs experiments) of this proposal's inliers are inside other active proposals, we identify this proposal as included by others. To seek a more abstract decomposition of the scene, we eliminate the proposals that are contained by others.

Number of proposal changes We observe that because of the efficient proposal management, the active proposals that we need to consider quickly decrease to a small number after a few iterations as shown in Fig. S2.

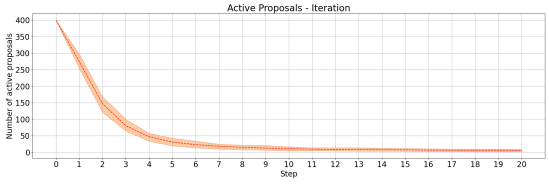

Figure S2. Number of active proposals at each step on the SynMugTree testset.

Visualization of iterations We also provide a per-step visualization in Fig S9.

S.2.3. Others

To boost the iteration, when querying the new assignment weight in the E-step, we only compute the query inside a large enough ball centered at the predicted center Θ_c instead of querying all the scene pointcloud.

We also sharpen the assignment weight by linearly increasing the α_D, α_N in Eq.6 in our main paper. For example, for the mug experiments, the starting values are $\alpha_D = 3.0$ and $\alpha_N = 0.003$ (with arccos returns in degree) and the ending values are $\alpha_D = 30.0$ and $\alpha_N = 0.006$ at the 10th step.

Another feature of our method is that we produce a wide range of proposals with low to high confidence, enabling the user to balance precision and recall. Note the mAP metrics will evaluate both higher precision and higher recall performance by computing the area of the Precision-Recall curve. We use the following confidence thresholds for the visual results in our main paper and supplementary document:

SynMugs Z	0.0	SynMugs SO3	0.0	SynMugs Pile	0.0
SynMugs Tree	0.4	SynMugs Box	0.7	SynMugs Shelf	0.3
RealMugs Z	0.0	RealMugs SO3	0.0	RealMugs Pile	0.2
RealMugs Tree	0.0	RealMugs Others	0.4	RealMugs Wild	0.3
SynChair Z	0.0	SynChair SO3	0.0	SynChair Pile	0.4
SynKit Z	0.2	SynKit SO3	0.2	SynKit Pile	0.2
RealChair Z	0.4	RealChair SO3	0.4	RealChair Pile	0.4
ScanNet	0.0				

Table S1. Visualization confidence thresholds

For the Scannet [6] experiments, since the scene is larger, we run EFEM for three rounds on each scene to greedily cover the objects. In later rounds, we mark the previous output points as invalid and the numbers of initialization proposals in the three rounds are $[300, 200, 100]$. Since ScanNet has no unusually oriented chairs, we also filter out the proposals where the predicted pose is not upright.

S.3. More Ablations and Studies

S.3.1. Shape Prior Network Design

The four components of the shape latent code Θ are necessary for the SIM(3)-equivariant shape reconstruction. Θ_s, Θ_c are for factoring out scales and translations (especially for stabilizing the partial observations); Θ_R and the inner product for rotations; and the invariant feature Θ_{inv} is for global shape conditioning following the VN-ONet in the original VNN [7]. Tab. S2 reports an ablation (SynMugTree) on using the centroid and radius of the input as Θ_c, Θ_s instead of predicting Θ_c, Θ_s by the network; and removing rotation equivariance by using a vanilla PointNet and trained with or without the rotation augmentation.

Setting	**AP**	**AP50**	**AP25**
original	68.8	99.0	99.8
No learned Θ_c, use centroid	65.7	99.0	99.5
No learned Θ_s, use radius	65.3	99.1	99.5
PointNet No-Augmentation	6.8	40.3	82.2
PointNet R-Augmentation	52.1	79.4	84.9

Table S2. Ablation study for equivariant shape prior network (equivariant shape priors in gray).

S.3.2. Different Initialization and timing

We further study how our performance and running time is affected by the initialization strategy in Tab. S3. We report the performance and the averaged inference time (per scene, all running on a laptop with i9-11950H CPU and RTX3080 GPU) with different initialization parameters on SynMugTree (number n and radius r of the initial proposals, original n=400,r=0.07). We also test on a different crop pattern with vertical cylinders for SynMugTree. Running the inference on SynMugTree 4 times with totally different random seeds leads to an mAP50 STD 0.53%.

Setting (r=0.07)	**AP**	**AP50**	**Time** (s)
original (n=400)	67.9	99.1	31.8
n=50	39.9	62.0	7.7
n=200	68.2	98.3	19.0
n=800	68.2	99.6	58.4
Setting (n=400)	**AP**	**AP50**	**Time** (s)
r=0.04	66.5	97.8	21.2
r=0.10	68.3	98.1	37.3
r=0.20	57.0	82.6	53.0
cylinder(r=0.07,h=∞)	67.9	99.1	31.4

Table S3. Study for initialization parameters and running time on SynMugTree

S.4. Dataset Details

S.4.1. Synthetic Scenes

We simulate our synthetic dataset with the SAPIEN [12] simulator to produce physically realistic arrangements of the objects.

Assets To create assets for our simulation, we first preprocess the corresponding category meshes from ShapeNet via the convex decomposition algorithm CoACD [11]. If no convex decomposition is applied, we can not generate physically plausible contact, such as mugs hanging on the tree or piles of mugs. We use the dataset split from our shape-prior training to generate different folds (train/val/test) of the scenes ensuring that all the objects in testing scenes are novel to our shape prior.

Tabletop Scenes For the synthetic tabletop scenes (Mugs and Kitchen containers), we place 4 synthetic depth cameras at the corners of a table and place the objects in a bin on the table, which is a common setup for tabletop manipulators. As demonstrated in Fig.3 in the main paper, we simulate realistic IR sensor depth patterns because we take advantage of the IR ray tracing [14] simulation in SAPIEN [12]. The mesh reconstruction is created by integrating 4 view depths via TSDF fusion. For the mugs **Shelf** configuration, we reduce the number of depth cameras from 4 to 2 since the shelf is only visible from one side as shown in Fig.3 in our main paper.

Chair Scenes Unlike tabletop scenes, real-world indoor scenes are always captured by continuous scans instead of static cameras. The continuous scan will result in smoother and better reconstruction. Therefore we capture the chair scene with 8 static cameras with ideal depth (not IR ray tracing). We use more cameras to capture more complete surfaces and since the depth is ideal, the integration from 8 static views can be regarded as similar to the quality of the scan.

S.4.2. Real Scenes: Chairs and Mugs

Our real dataset contains 240 reconstructions of real scenes containing challenging configurations and backgrounds. The distribution of configurations is given in Table S4. More data are collected for scenes with more complex configurations or that are harder to create in simulation environments.

RealMugs Z	10	RealMugs SO3	10	RealMugs Pile	10
RealMugs Tree	50	RealMugs Others	50	RealMugs Wild	50
RealChair Z	20	RealChair SO3	20	RealChair Pile	20

Table S4. Chairs and Mugs dataset counts

Collection As shown in Fig.S6, the tabletop data collection setup is very similar to the simulation. We mount 4 realsense D455 on 4 corners of the table. The poses of the cameras are calibrated manually by a chessboard in the middle of the table. For the real mugs **Others** configuration, we use random distractor objects. For the real mugs **Wild** configuration and all chair scenes (Fig. S7), we use an iPad with a lidar scanner to scan the wild scenes.

Annotation We develop a lightweight annotation tool based on MeshLab [5]. All the chairs and mugs in the scene are annotated with the instance ID at each vertex.

S.5. Additional Results

S.5.1. Comparison to Baselines

We show additional comparisons between our method and the baselines on the real-mug **Tree**, **Others**, and **Wild** test sets while trained on different synthetic setups. Fig. S3 shows the precision and recall of different methods, where the widely adopted mAP metric is the area under the Precision-Recall curve (leftmost in Fig. S3). Ours is displayed in dark blue, which shows advantages over other baselines.

We also show visual results of ours and all the baselines on real mugs **Others** (Fig. S5) and **Wild** (Fig. S4) test set. Since the visual results are affected by the confidence selection, we pick a threshold that has relatively better precision by the PR curve Fig S3 for all the methods and show both the zero threshold (all predicted masks) on the left column in Fig. S5,S4 and the selected threshold on the right column.

S.5.2. More visualization on Chairs and Mugs

We show more visual results on our Chairs and Mugs real-world test set in Fig. S6 and Fig S7.

S.5.3. More results on ScanNet

We show more qualitative results on ScanNet in Fig. S8. And we further report Tab. S5 for comparison with baselines on ScanNet testset chairs category. Note that some failure cases are also shown in Fig. S8, where some other stuff that looks like a chair (e.g. the basin in a rest room, right bottom corner in Fig. S8) are also recognized as the chair since our model is fully geometry based. Some chairs in ScanNet are also not detected since they are too partial. We leave this gap for future works to fill.

Method	**AP**	**A50**	**A25**
EFEM (None)	20.2	39.0	48.3
CSC (20pts)	26.4	42.4	56.3
CSC (200pts)	41.5	61.1	70.4
Box2Mask (Box)	62.5	81.6	88.7
SoftGroup (Full)	69.4	86.2	91.3

Table S5. Comparison with baselines on ScanNet testset chairs category.

S.6. Full tables

We provide the full mAP25% here due to the page limit in the main paper. We note that in some novel scenes, the weakly supervised Box2Mask [4] works even better than the supervised SoftGroup [9]. Potential reasons might be: (1.) Box2Mask uses over-segmentation as input, which provides more structural information and is more robust in novel scenes. (2.) the in-accurate box supervision provides some augmentation that makes the networks harder to overfit.

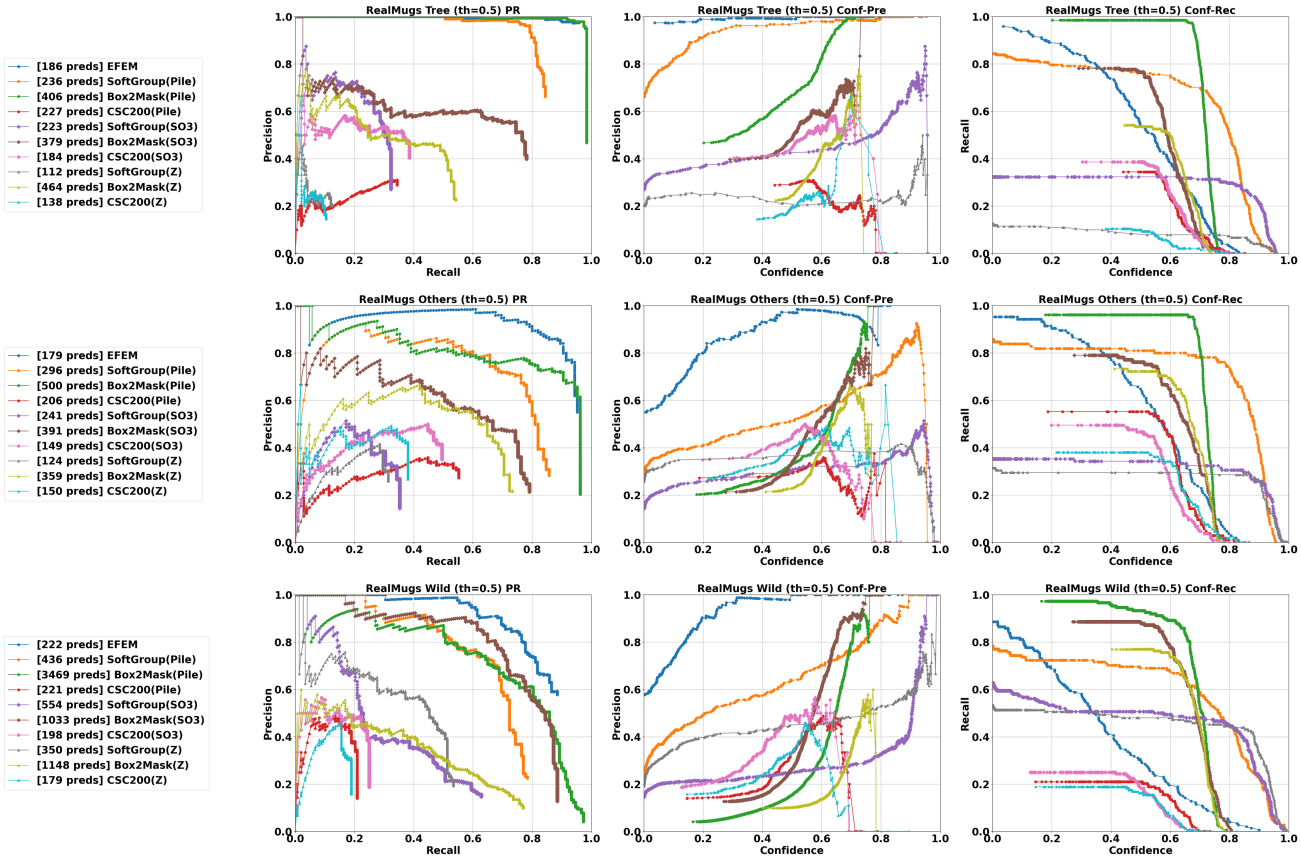

Figure S3. Precision-Recall (IoU50%) on RealMugs novel settings. In the legend on the left, the setting names in the bracket indicate where the baselines are trained and the number of predicted masks over the whole testing set is reported on the left. Note that Box2Mask [4] predicts large numbers of low confidence masks in the **Wild** setup.

SynMugs	Testing	Z			SO3			Pile			Tree			Box			Shelf		
Training	**Metrics**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**
Scene Z	CSC (100)	6.0	22.0	49.3	1.9	7.6	26.4	0.1	0.5	5.8	0.0	0.0	1.3	0.0	0.2	2.2	0.0	0.0	0.8
	CSC (200)	78.7	98.1	98.1	62.7	83.3	88.6	8.9	17.4	35.6	0.6	2.0	18.1	2.0	5.0	11.8	0.8	3.1	21.2
	Box2Mask	96.9	**100**	**100**	92.1	99.3	99.6	36.7	27.6	84.6	12.3	45.3	80.9	10.0	24.0	33.8	8.6	14.6	16.5
	SoftGroup	**100**	**100**	**100**	96.5	98.5	98.5	21.9	27.6	35.6	0.4	1.8	10.9	1.6	3.7	20.3	9.2	17.2	35.9
Scene SO3	CSC (100)	1.7	8.6	44.3	1.4	6.5	29.1	0.1	0.6	5.6	0.0	0.0	2.4	0.0	0.1	2.1	0.1	0.6	5.6
	CSC (200)	70.0	99.3	99.4	72.7	95.7	97.9	22.3	42.2	60.7	10.2	24.6	48.7	5.4	12.3	21.1	5.2	20.0	61.2
	Box2Mask	95.8	**100**	**100**	94.7	**100**	**100**	53.7	80.1	91.9	30.2	80.6	92.5	26.5	40.1	45.4	13.4	33.0	51.2
	SoftGroup	**100**	**100**	**100**	99.6	99.8	99.8	43.9	51.9	58.4	10.0	18.7	28.7	15.3	21.3	23.8	20.8	30.9	32.0
Scene Pile	CSC (100)	0.0	0.3	6.0	0.0	0.3	5.8	0.0	0.1	4.2	0.0	0.0	1.1	0.0	0.0	0.8	0.0	0.0	6.0
	CSC (200)	53.3	88.4	91.9	50.2	77.7	84.1	29.5	61.3	76.3	29.4	66.2	81.3	7.6	19.8	27.1	10.3	34.1	**84.4**
	Box2Mask	96.0	**100**	**100**	94.7	**100**	**100**	78.7	**99.6**	**99.9**	55.2	94.8	96.1	42.7	52.9	53.0	26.3	54.6	67.2
	SoftGroup	99.5	**100**	**100**	**99.7**	**100**	**100**	**89.0**	93.2	93.6	42.3	72.8	76.3	23.4	25.5	25.7	28.8	39.5	42.6
ShapeNet	EFEM	78.4	99.8	**100**	79.3	99.8	**100**	68.2	96.8	97.6	**68.8**	**99.0**	**99.8**	**59.9**	**77.0**	**77.6**	**48.7**	**72.4**	80.7

Table S6. Full table with mAP25% on SynMugs test set.

Figure S4. Comparison on RealMugs **Wild** test set. The left labels show the name of the methods and where they are trained. The two numbers below the method names are the visualization thresholds. The left column is with threshold 0.0, which means showing all the predicted masks; and the right column is with the thresholds we picked from the precision-recall curves (Fig. S3).

Figure S5. Comparison on RealMugs **Others** test set. The format is the same as Fig. S4.

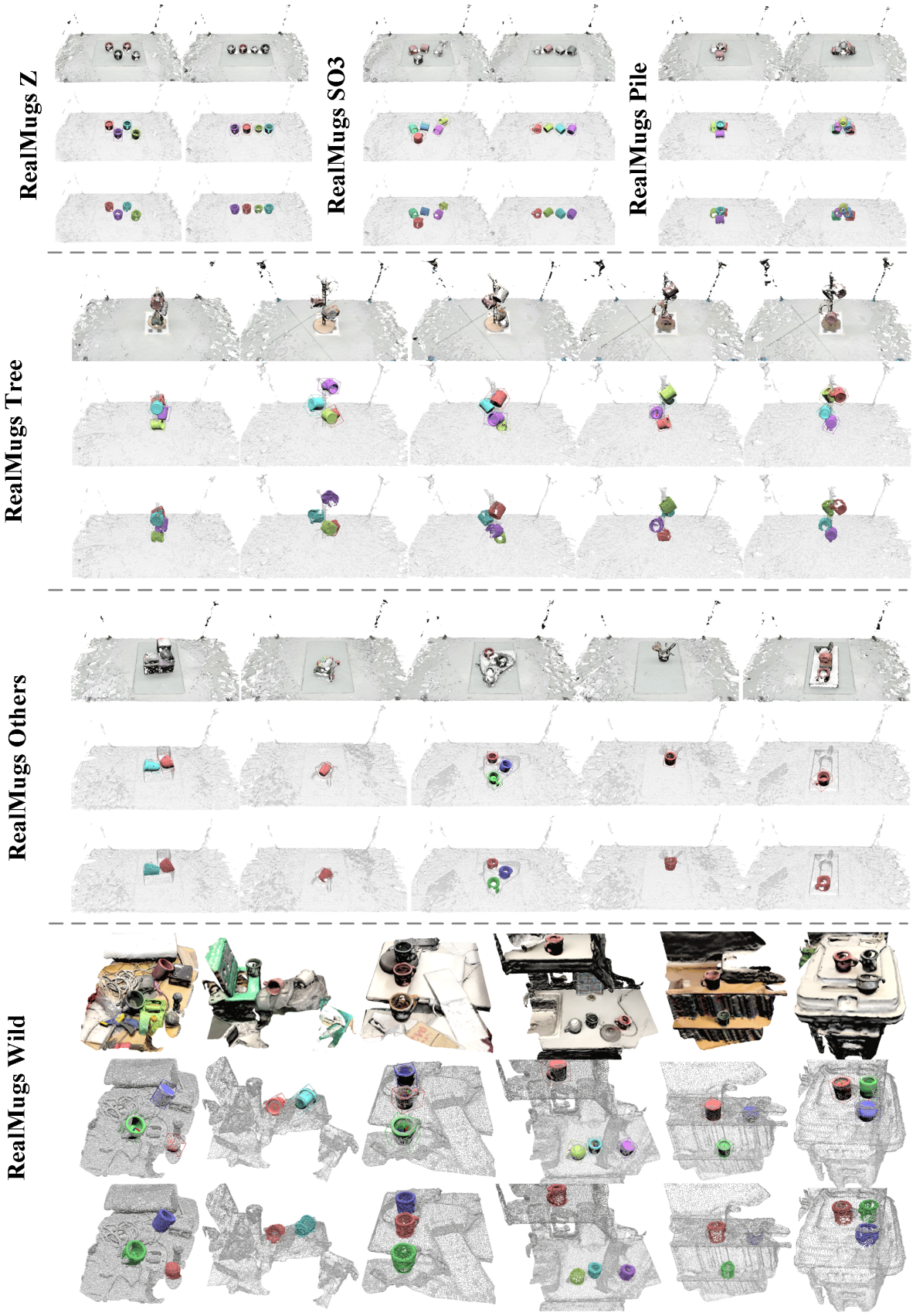

Figure S6. More visual results on RealMugs. We show the shape, pose, and bounding box by-products on the second row and the predicted masks on the third row.

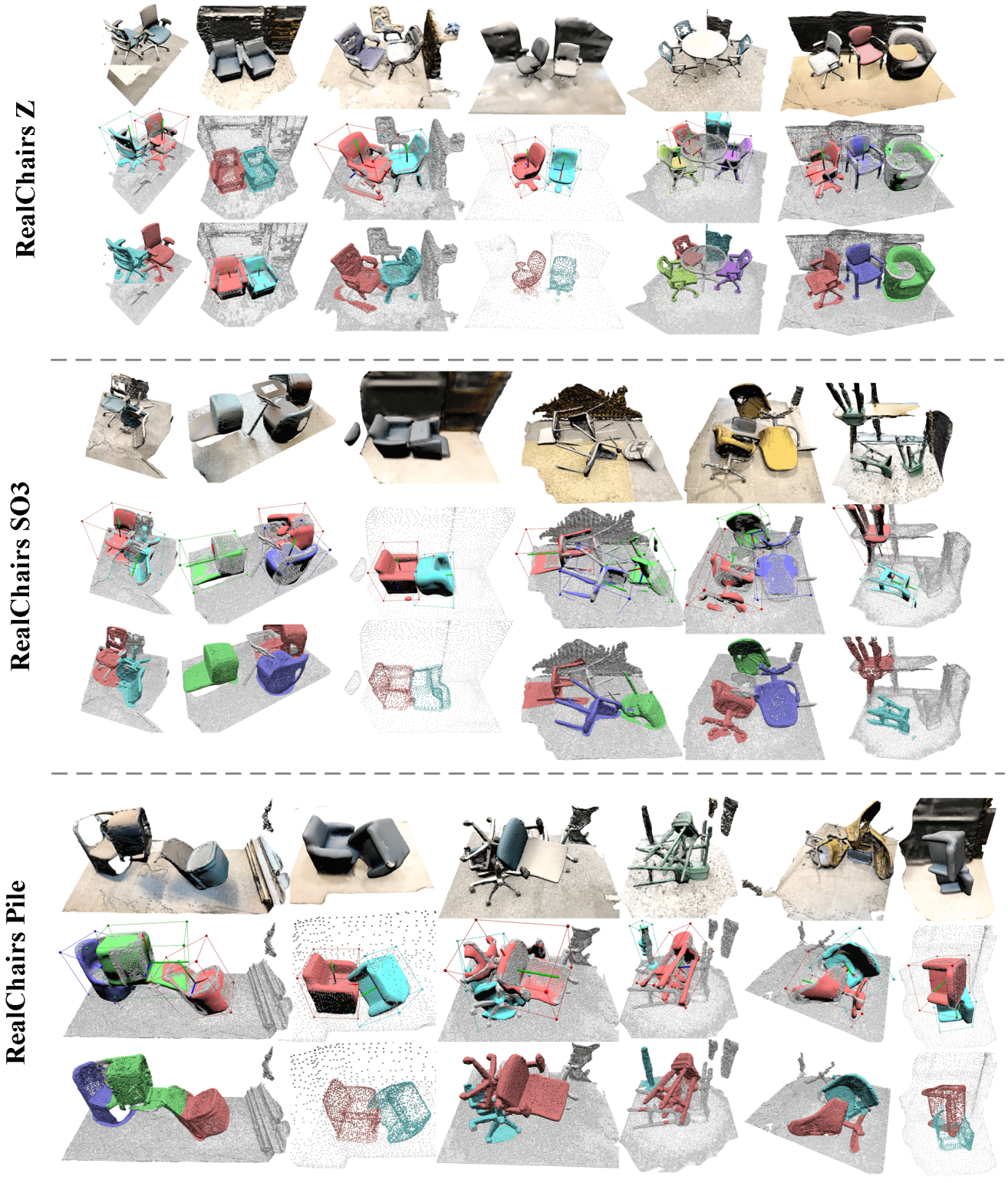

Figure S7. More visual results on RealChairs. The format is the same as Fig. S6.

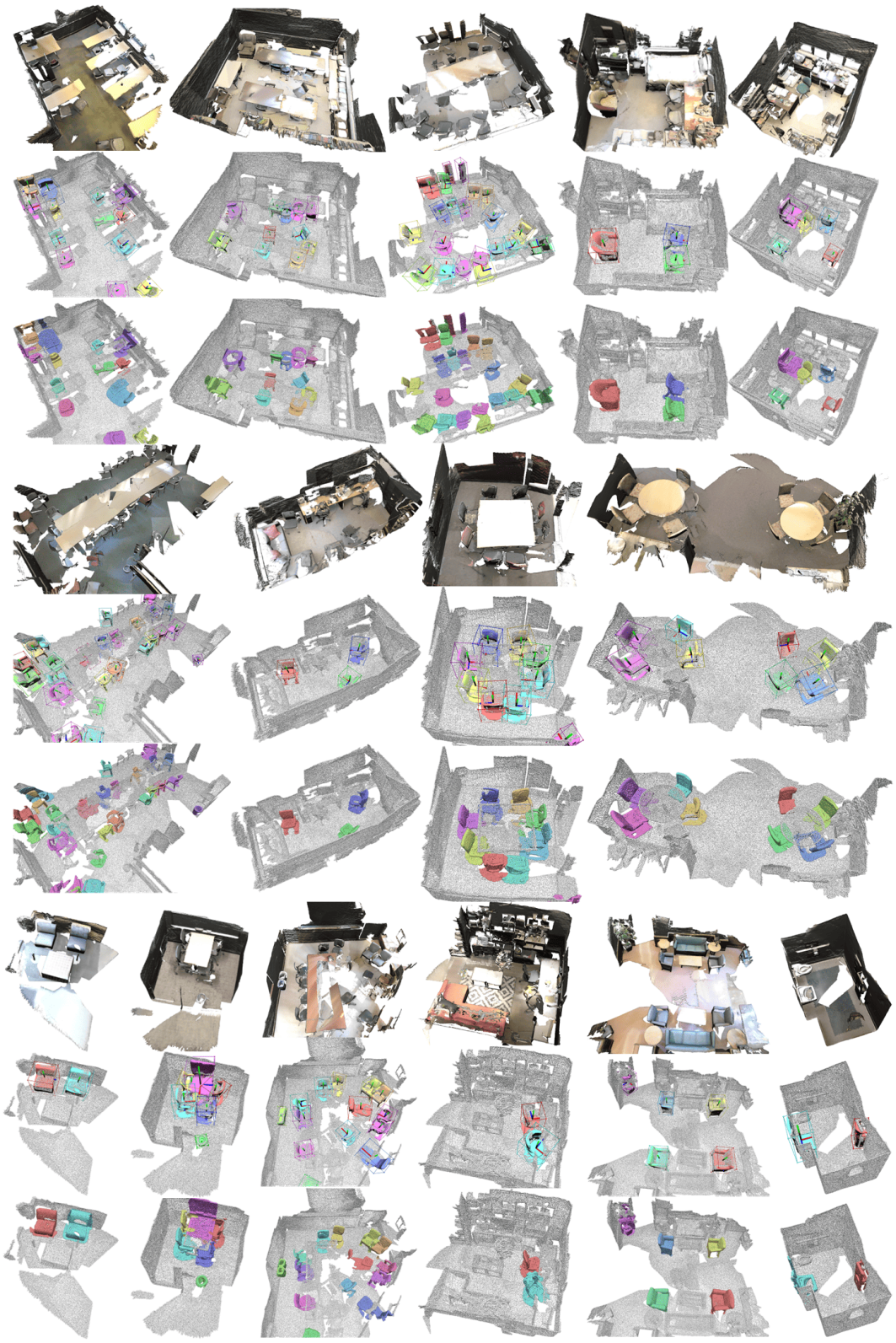

Figure S8. More visual results on ScanNet.The format is the same as Fig. S6.

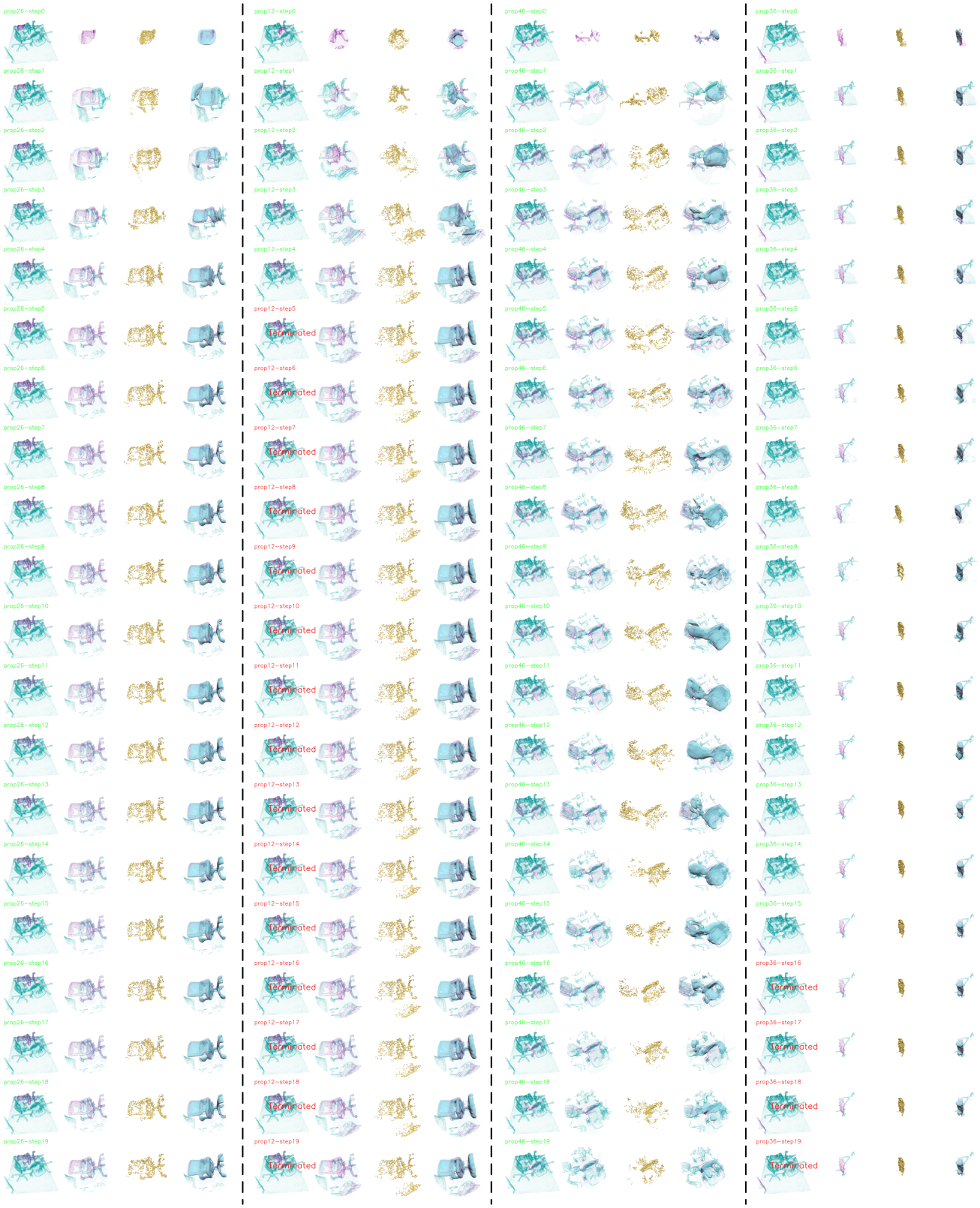

Figure S9. Visualization of all steps of the 4 proposals shown in the main method figure (Fig.2) in our paper. From left to right corresponds to the first to the fourth rows in Fig.2 in our main paper. The left top corner has the step number and if the proposal is terminated, the proposal will be marked in red. Each step has 4 sub-figure visualizations, from the left to the right: (1.) the visualization of assignment mask W_t on the full point cloud; (2.) a local crop of W_t; (3.) the point cloud P_t sampled based on W_t; (4.) the estimated shape based on the sampled point cloud P_t.

RealMugs	Testing	Real Z (10)			Real SO3 (10)			Real Pile (10)			Real Tree (50)			Real Others (50)			Real Wild (50)		
Training	**Metrics**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**
Scene Z	CSC (100)	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.2	0.9	1.8
	CSC (200)	72.1	99.8	99.8	9.5	21.4	44.6	0.3	1.6	11.5	0.6	2.8	19.7	6.7	16.0	41.1	2.0	6.1	14.7
	Box2Mask	98.1	**100**	**100**	93.1	**100**	**100**	5.6	19.1	72.9	8.4	28.8	71.6	18.4	39.5	73.2	18.1	27.9	30.9
	SoftGroup	93.7	97.6	97.6	97.3	**100**	**100**	0.0	0.0	8.2	0.9	3.4	28.5	4.3	9.1	50.0	18.3	32.9	65.7
Scene SO3	CSC (100)	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.7	1.9	2.9
	CSC (200)	68.4	**100**	**100**	59.8	**100**	**100**	1.0	3.7	23.1	4.8	20.1	57.9	6.9	19.7	47.9	3.9	11.8	34.2
	Box2Mask	**100**	**100**	**100**	95.3	**100**	**100**	12.7	34.3	74.5	19.6	49.2	67.8	27.1	50.6	73.8	50.3	74.8	**82.4**
	SoftGroup	99.2	**100**	**100**	98.2	**100**	**100**	2.8	4.8	25.5	10.4	21.7	45.4	6.5	13.3	48.3	18.3	30.7	41.8
Scene Pile	CSC (100)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	1.8	3.5
	CSC (200)	13.0	45.0	64.2	22.7	65.2	78.3	5.3	30.1	82.5	1.5	8.0	53.6	3.5	15.4	45.3	2.8	8.5	23.3
	Box2Mask	**100**	**100**	**100**	95.8	**100**	**100**	72.3	**98.0**	**100**	55.8	**98.1**	**99.6**	56.0	79.2	81.4	45.1	71.4	73.6
	SoftGroup	99.6	**100**	**100**	**99.7**	**100**	**100**	**74.7**	86.3	86.3	53.4	83.0	88.5	50.9	66.5	68.8	48.1	65.9	69.4
ShapeNet	EFEM	87.2	**100**	**100**	87.3	**100**	**100**	63.1	94.0	94.0	**69.6**	95.4	96.2	**69.7**	**89.4**	**90.0**	**54.1**	**82.3**	82.3

Table S7. Full table with mAP25% on RealMugs testset.

SynChairs	Testing	Z			SO3			Pile		
Training	**Metrics**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**
Scene Z	CSC (100)	66.2	83.1	85.7	56.6	80.2	81.9	6.1	15.2	37.1
	CSC (200)	77.7	91.0	92.8	63.7	85.1	86.6	10.5	22.7	44.7
	Box2Mask	**99.9**	**100**	**100**	95.8	98.9	98.9	19.3	41.2	73.4
	SoftGroup	99.8	**100**	**100**	94.8	98.5	98.5	24.1	36.5	53.2
Scene SO3	CSC (100)	74.9	81.4	84.1	77.3	82.4	83.3	17.7	31.0	45.4
	CSC (200)	84.3	86.6	87.4	85.4	88.5	89.2	20.6	35.3	49.6
	Box2Mask	99.7	**100**	**100**	99.1	99.5	**100**	50.5	80.3	92.2
	SoftGroup	99.6	**100**	**100**	99.1	**100**	**100**	55.1	68.2	75.1
Scene Pile	CSC (100)	67.0	74.1	78.7	74.6	82.1	85.3	47.2	63.8	67.9
	CSC (200)	71.7	76.3	79.3	88.4	92.4	94.5	67.4	81.6	84.6
	Box2Mask	99.6	99.7	99.7	**99.5**	99.7	**100**	78.6	96.9	**98.4**
	SoftGroup	99.4	**100**	**100**	98.9	**100**	**100**	**95.0**	**97.0**	97.1
ShapeNet	EFEM	93.1	99.2	99.5	86.1	97.4	97.5	75.3	88.0	94.6

Table S8. Full table with mAP25% on SynChairs testset.

SynKit	Testing	Z			SO3			Pile		
Training	**Metrics**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**	**AP**	**AP50**	**AP25**
Scene Z	CSC (100)	12.7	29.9	49.5	7.5	14.5	25.8	2.3	5.9	15.4
	CSC (200)	63.0	80.9	87.3	37.9	49.2	69.3	12.2	23.2	48.8
	Box2Mask	89.7	97.3	**98.6**	69.6	87.0	91.8	41.4	68.9	84.6
	SoftGroup	93.6	96.8	97.2	94.0	99.3	**99.5**	52.3	63.5	72.8
Scene SO3	CSC (100)	8.5	21.7	40.7	11.3	23.8	55.8	2.0	5.4	21.4
	CSC (200)	27.8	48.9	66.0	53.9	77.5	87.7	18.1	38.9	62.9
	Box2Mask	85.3	96.2	96.8	91.8	98.5	**99.5**	52.9	76.5	87.8
	SoftGroup	89.3	93.9	94.5	**96.9**	**99.3**	99.3	56.2	67.2	72.9
Scene Pile	CSC (100)	3.4	11.2	24.0	2.6	7.1	20.1	1.4	4.5	19.1
	CSC (200)	26.5	53.8	72.8	44.8	73.6	90.2	28.8	60.4	81.2
	Box2Mask	87.2	**97.9**	98.1	92.1	98.6	99.4	74.8	**94.6**	**97.0**
	SoftGroup	**93.7**	97.3	97.7	96.0	98.8	**99.5**	**84.6**	91.3	92.0
ShapeNet	EFEM	69.4	83.4	83.7	67.6	83.1	84.6	60.1	78.9	80.7

Table S9. Full table with mAP25% on SynKit testset.

References

[1] Serge Assaad, Carlton Downey, Rami Al-Rfou, Nigamaa Nayakanti, and Ben Sapp. Vn-transformer: Rotation-equivariant attention for vector neurons. *arXiv preprint arXiv:2206.04176*, 2022. 1, 2, 3

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 4

[3] Yunlu Chen, Basura Fernando, Hakan Bilen, Matthias Nießner, and Efstratios Gavves. 3d equivariant graph implicit functions. *ECCV*, 2022. 2, 3

[4] Julian Chibane, Francis Engelmann, Tuan Anh Tran, and Gerard Pons-Moll. Box2mask: Weakly supervised 3d semantic instance segmentation using bounding boxes. In *European Conference on Computer Vision*, pages 681–699. Springer, 2022. 7, 8

[5] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 7

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 6

[7] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021. 1, 2, 3, 6

[8] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[9] Thang Vu, Kookhoi Kim, Tung M Luu, Thanh Nguyen, and Chang D Yoo. Softgroup for 3d instance segmentation on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2708–2717, 2022. 7

[10] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 1

[11] Xinyue Wei, Minghua Liu, Zhan Ling, and Hao Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *arXiv preprint arXiv:2205.02961*, 2022. 6

[12] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020. 6, 7

[13] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. *Advances in Neural Information Processing Systems*, 32, 2019. 4

[14] Xiaoshuai Zhang, Rui Chen, Fanbo Xiang, Yuzhe Qin, Jiayuan Gu, Zhan Ling, Minghua Liu, Peiyu Zeng, Songfang

Han, Zhiao Huang, Tongzhou Mu, Jing Xu, and Hao Su. Close the Visual Domain Gap by Physics-Grounded Active Stereovision Depth Sensor Simulation. *arXiv preprint*, 2022. 7

[15] Minghan Zhu, Maani Ghaffari, and Huei Peng. Correspondence-free point cloud registration with so (3)-equivariant implicit shape representations. In *Conference on Robot Learning*, pages 1412–1422. PMLR, 2022. 5