Spatially Invariant Unsupervised 3D Object-Centric Learning and Scene Decomposition –Supplementary Material–

Tianyu Wang[1][0000−0001−9032−8488], Miaomiao Liu[1][0000−0001−6485−3510], and Kee Siong Ng[1][0000−0003−0701−8783]

Australian National University, Canberra ACT 2601, AU
{Tianyu.Wang2,miaomiao.liu,KeeSiong.ng}@anu.edu.au

1 Computational Cost Analysis

In this section, we provide an example of time complexity reduction induced by the local glimpse proposal scheme. Generative models without the local glimpse proposal scheme span each mixture component across the entire observation [1][3][4][8]. For point cloud data, this formulation leads to a huge computational cost. In particular, the computational cost is mainly from the computation of the Chamfer mixture loss, which requires the bidirectional closest point search between the ground truth and the estimated point cloud. For a scene consisting of 10,000 points, the mixture loss requires the computation of distances between 50 million (50,000,000) unique pairs of points per component. By contrast, the local glimpse proposal scheme reduces the time complexity by confining the matching computation within each glimpse since the weights for points outside of the glimpse are set as 0. In general, a glimpse containing 1,000 points (10% of 10,000) leads to 500k (1% of 50,000,000) unique pairs of points, which is much less compared with the computation on the entire point cloud. Thus, with non-overlapping voxel grid cells, the total time complexity should be much lower compared to a full-scope mixture component, which is the key to scalability. The exact time complexity reduction is determined by both glimpse size and point density.

2 Hyperparameters and Prior Distributions

In this section, we present hyperparameters and the prior distributions of latent variables. We use a quadruple (n, m, p, q) to denote annealing of hyperparameter values from n to m, starting from iteration p to iteration q. Glimpse-related hyperparameters are shown in Table 1. Priors are specified in Table 2. Other hyperparameters are specified in Table 3.

Term	Value
spatial attention grid cell size	1
glimpse max apothem	1
glimpse min apothem	0.25
glimpse max center offset	0.75

Table 1: Glimpse related hyperparameters.

Term	Value
$\mu^{apothem}$	$(2, -1, 10000, 20000)$
β^{pres}	$(0.01, 0.0001, 0, 15000)$
z^{where} prior	$\mathcal{N}(0, 0.5)$
$z^{apothem}$ prior	$\mathcal{N}(\mu^{apothem}, 0.5)$
z^{pres} prior	$Bernoulli(\beta^{pres})$
z^{what} prior	$\mathcal{N}(0, 1)$
z^{mask} prior	$\mathcal{N}(0, 1)$

Table 2: Prior distributions.

Term	Value
CML σ_c	$(0.1, 0.05, 10000, 15000)$
CML dist.	$\mathcal{N}(0, \sigma_c)$
relaxed Bernoulli temp	$(2.5, 0.5, 0, 10000)$
PGD initial distribution	$\mathcal{N}(0, 0.3)$

Table 3: Other hyperparameters.

Mixing Weight. Recall that the mixing weight defined in our main paper for the *forward Chamfer Likelihood* implies the segmentation of the point cloud. To encourage the mixing weight to approach 0 or 1, we include a temperature factor 10 into the computation of α_i^x, and the mixing weight is implemented as

$$\alpha_i^x = \frac{(z_i^{pres}\pi_i^x)^{10}}{\sum_{j=1}^{n}(z_j^{pres}\pi_j^x)^{10}} z_i^{pres}\pi_i^x. \tag{1}$$

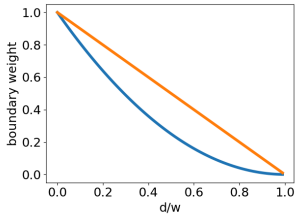
Weights on KL Divergence. Here, we introduce the detailed formulation of $\mathcal{L}_{KL}$, the second term of the evidence lower bound, in the following.

$$\begin{aligned}\mathcal{L}_{KL}(\mathbf{z}^{cell}, \mathbf{z}^{object}, \mathbf{z}^{scene}) = w\mathcal{D}_{KL}(p(z^{pres})||q(z^{pres}|x)) + z^{pres}[\\ \mathcal{D}_{KL}(p(\mathbf{z}^{what})||q(\mathbf{z}^{what}|x))+ \\ \mathcal{D}_{KL}(p(\mathbf{z}^{mask})||q(\mathbf{z}^{mask}|x))+ \\ \mathcal{D}_{KL}(p(\mathbf{z}^{where})||q(\mathbf{z}^{where}|x))+ \\ \mathcal{D}_{KL}(p(\mathbf{z}^{apothem})||q(\mathbf{z}^{apothem}|x))].\end{aligned} \tag{2}$$

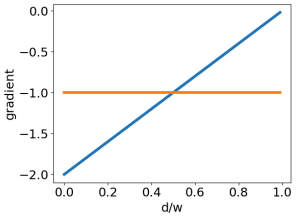
Note that the weight w for KL Divergence of z^{pres} is to encourage glimpse rejection. More specifically, w is set to 10 for UOR dataset and is annealed from 10 to 20 in the first 15000 steps for UOT dataset in our experiments. Following the SPAIR model [2], we also set the weight for KL divergence of the rest latent variables with z^{pres} so that the rejected glimpses won't produce penalties encouraging glimpse rejection.

Training. We use Adam [6] optimizer with the learning rate set to 0.0001 during our training process.

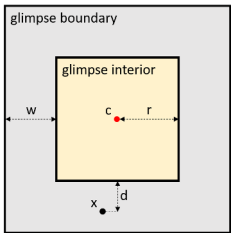
3 Soft Boundary



(a) Weight function.



(b) Gradient.



(c) Glimpse boundary structure.

Fig. 1: Visualization of glimpse boundary structure, glimpse boundary weights and the corresponding gradients. Fig. 1c illustrates the glimpse structure where c is the glimpse center and x is an point living in the glimpse boundary. The linear decay function (orange) and parabola decay function (blue) are plotted in Fig. 1a with the corresponding gradients shown in Fig. 1b.

As shown in Fig 1c, we divide a glimpse into glimpse interior of apothem r and glimpse boundary of width w. For an input point x in glimpse $\mathcal{G}_i$ with center location c_i, its boundary weight $b_i^x \in (0, 1]$ is defined as

$$b_i^x = \begin{cases} 1, & x \in \text{glimpse interior}; \\ f(d, w), & x \in \text{glimpse boundary}, \end{cases}$$

where f is a continuous function and $d = \|x - c_i - r\|_{\text{inf}}$.

One example of f is a linear decay function $f(d, w) = 1 - \frac{d}{w}$. Another example is a parabola decay function $f(d, w) = (\frac{d}{w})^2 - 2(\frac{d}{w}) + 1$, which generates a larger gradient compared with the linear one when $d \leq \frac{w}{2}$, and a smaller gradient, otherwise. For the case where the distance between two objects is smaller than w but larger than $\frac{w}{2}$, the parabola decay function is preferred to prevent overly large glimpses. In this work, we use the parabola decay function and set the width of the glimpse boundary as $w = 0.75r$.

From Fig. 1a and Fig. 1b, we observe that both types of boundary decay function f provide a negative gradient when points are being excluded (increasing $\frac{d}{w}$ value) from a glimpse.

4 Model Structure

In this section, we introduce the detailed model structure. To make our work self-contained we briefly introduce the PointConv layer [10] and PointGNN layer [9] below. PointConv generalizes convolution operation from discrete domain to continuous domain. One PointConv layer is specified with (c_{mid}, c_{out}) (See Fig. 2 for structure illustration). To make PointConv invariant of the total number of input points, we divide the output feature value by the total number of input points.

One PointGNN layer contains three two-layer MLPs which are MLP_h, MLP_f and MLP_g, respectively and the structure can be summarized by

$$s_i^{out} = g(\max_{j \in \mathcal{N}_i}\{f(x_j - x_i + h(s_i^{in}), s_i^{in})\}, s_i^{in}), \tag{3}$$

where s_i^{in} and s_i^{out} are the features of the point i before and after PointGNN layer, $\mathcal{N}_i$ defines the points connected to i, x_i indicates the 3D coordinate of the point i. The max operation is performed over all points j that are connected to the point i. For all PointGNN layers, we consistently set $h_{hidden} = 32$ and $h_{out} = 3$. Thus, we define the structure of a PointGNN layer with parameters as $(f_{hidden}, f_{out}, g_{hidden}, g_{out})$. We also list the parameters for other operations such as the radius for radius graph operation, the voxel cell size for voxel pooling operation, and the output size for linear layers in following tables. Table 10 shows the structure of voxel grid encoder. The structure of glimpse VAE including the glimpse encoder, the mask decoder, the glimpse Point Graph Flow, and the Multi-layer PointGNN is presented in Table 4, Table 7, Table 6 and Table 9, respectively. The structure of global VAE including the Global encoder and the Global Point Graph Flow is detailed in Table 8 and Table 5. Input point coordinates are reduced by a factor of 16.

Layer/Operation	Parameter
Radius Graph	0.25
PointGNN	$(8, 8, 8, 8)$
LayerNorm	
Voxel Pool	0.25
PointConv	$(16, 32)$
Celu	
Radius Graph	0.5
PointGNN	$(32, 32, 32, 32)$
LayerNorm	
Voxel Pool	0.25
PointConv	$(64, 128)$
Celu	
Radius Graph	1.0
PointGNN	$(128, 128, 128, 128)$
LayerNorm	
PointConv	$(128, 256)$
Celu	
Linear	256

Table 4: Glimpse encoder.

Layer/Operation	Parameter
Random Sampling	
Radius Graph	0.2
PointGNN	$(128, 128, 128, 64 + 3)$
Radius Graph	0.1
PointGNN	$(64, 64, 64, 32 + 3)$
Radius Graph	0.05
PointGNN	$(16, 16, 16, 3)$

Table 5: Global Point Graph Flow.

Layer/Operation	Parameter
Random Sampling	
Radius Graph	0.2
PointGNN	$(128, 128, 128, 64 + 3)$
Radius Graph	0.1
PointGNN	$(64, 64, 64, 32 + 3)$
Radius Graph	0.05
PointGNN	$(16, 16, 16, 3)$

Table 6: Glimpse Point Graph Flow.

Layer/Operation	Parameter
PointConv	$(64, 32)$
Celu	
PointConv	$(16, 16)$
Celu	
PointConv	$(8, 8)$
Celu	
Linear	1

Table 7: Mask decoder.

Layer/Operation	Parameter
Radius Graph	0.25
PointGNN	$(8, 8, 8, 8)$
LayerNorm	
Voxel Pool	0.25
PointConv	$(16, 32)$
Celu	
Radius Graph	0.5
PointGNN	$(32, 32, 32, 32)$
LayerNorm	
Voxel Pool	0.25
PointConv	$(64, 128)$
Celu	
Radius Graph	1.0
PointGNN	$(128, 128, 128, 128)$
LayerNorm	
PointConv	$(128, 256)$
Celu	
PointConv	$(256, 512)$

Table 8: Global encoder.

Layer/Operation	Parameter
Random Sampling	
Radius Graph	1.0
PointGNN	$(128, 64, 64, 64)$
PointGNN	$(32, 32, 32, 32)$
PointGNN	$(16, 16, 16, 8)$
Linear	1

Table 9: Multi-layer PointGNN.

Layer/Operation	Parameter
Radius Graph	0.0625
PointConv	(8, 8)
Celu	
PointConv	(16, 16)
Celu	
PointConv	(32, 32)
Celu	
Voxel Pool	0.03125
PointConv	(32, 64)
Celu	
Radius Graph	0.03125
PointGNN	(64, 64, 64, 64)
LayerNorm	
Voxel Pool	0.0625
PointConv	(64, 128)
Celu	
Radius Graph	0.125
PointGNN	(128, 128, 128, 128)
LayerNorm	
Voxel Pool	0.125
PointConv	(128, 256)
Celu	
Radius Graph	0.25
PointGNN	(256, 256, 256, 256)
PointGNN	(256, 256, 256, 256)
PointGNN	(256, 256, 256, 256)
LayerNorm	
PointConv	(256, 256)
Linear	12

Table 10: Voxel grid encoder.

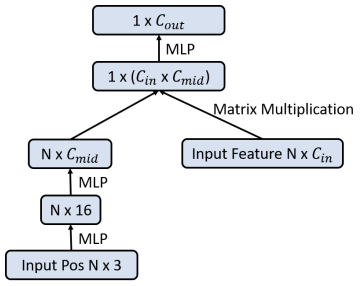


Fig. 2: Structure of PointConv.

5 Dataset Spec

We provide more details about the UOR and UOT dataset in this section. For both datasets, in each scene, 2-5 objects are uniformly randomly selected (with replacement) from the candidate set and placed at random locations in the scene with the constraint that they cannot largely overlap with each other (slight overlapping and touching are permitted). All objects are randomly rotated along y-axis.

For the point cloud construction, we convert the 10 depth frames for each scene into 10 partial point clouds. We then merge the 10 partial point clouds into a complete scene point cloud. To down sample the point clouds, we apply voxel grid pooling with cell size 0.15 on the scene point clouds. For all points in one cell, the coordinates are aggregated by average pooling operation.

The intrinsic parameters of cameras are shown in Table 11. The object pool for UOR and UOT are presented in Table 12 and Table 13, respectively, where we also specify the detailed dimension range of each object.

Term	Value
focal length	10 mm
sensor size x	16 mm
sensor size y	16 mm
clipping plane	20 m

Table 11: Camera intrinsic parameters

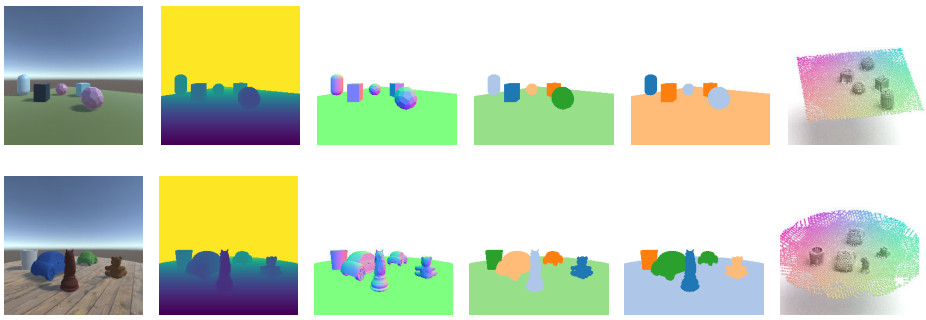


Fig. 3: Data captured by each camera in UOR dataset (top) and UOT dataset (bottom). From left to right are RGB, depth, normal, instance label, semantic label and constructed point cloud. Point clouds are obtained by merging multi-view depth images. Instance labels and semantic labels are used to train PointGroup [5] baseline. RGB images and normal maps are not used in this work.

To show that our dataset is indeed challenging for models capturing spatial structure correlations, we plot the empirical **closest neighbor distance distribution** of our data generation process. To obtain the distribution, for each object in each scene, we note down the distance between the **center** of this object and the **center** of its closest neighbor (the surface to surface distance is hard to compute). Thus, distance zero means a complete overlapping between two objects, which is not physically plausible. In our UOR and UOT dataset the minimum distance is set to one (not applicable to Object Matrix layout). With object dimensions specified in Table 12 and Table 13, we consider distance below 2 to be extremely close. Thus, reading from Fig.4, for scenes containing 2-5 objects, 25 percent of objects are spawned close to at least one other object. For scenes containing 6-12 objects, the number goes up to more than 60 percent.

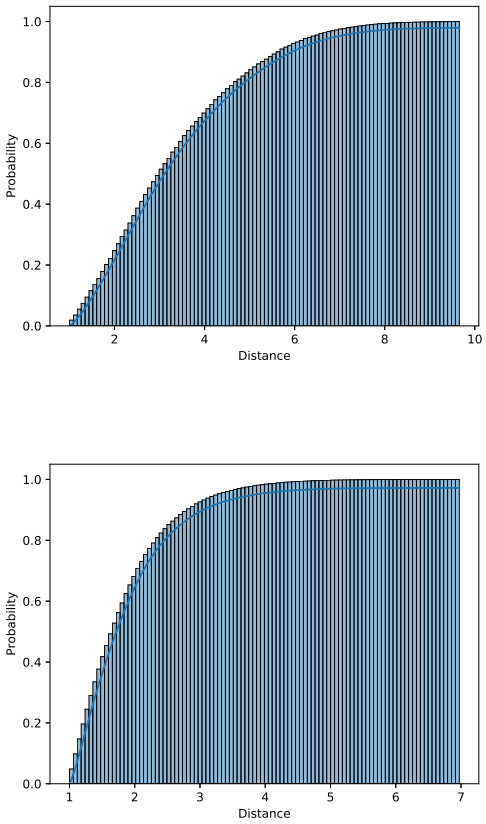


Fig. 4: Closest neighbor distance distribution of our data generation process for 2-5 objects (above), and for 6-12 objects (below)

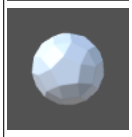
geometry	figure	min length/max length	min width/max width	min height/max height
Capsule (standing)		0.75/1.25	0.75/1.25	1.5/**2.5**
Capsule (flat)		1.5/**2.5**	1.5/**2.5**	0.75/1.25
Cube		0.75/1.25	0.75/1.25	0.75/1.25
Cylinder (standing)		0.75/1.25	0.75/1.25	1.5/**2.5**
Cylinder (flat)		1.5/**2.5**	1.5/**2.5**	0.75/1.25
Hexagonal Prism		1/1.33	1/1.33	0.5/0.83
Sphere		0.75/1.25	0.75/1.25	0.75/1.25
Rhombicosidodecahedron		0.75/1.25	0.75/1.25	0.75/1.25
Square Antiprism		1/1.5	1/1.5	0.67/1
Triangular Prism (standing)		0.75/1.5	0.75/1.5	0.75/1.5
Triangular Prism (flat)		0.75/1.5	0.75/1.5	0.75/1.5

Table 12: UOR object pool.

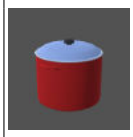
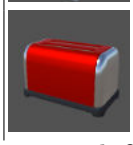
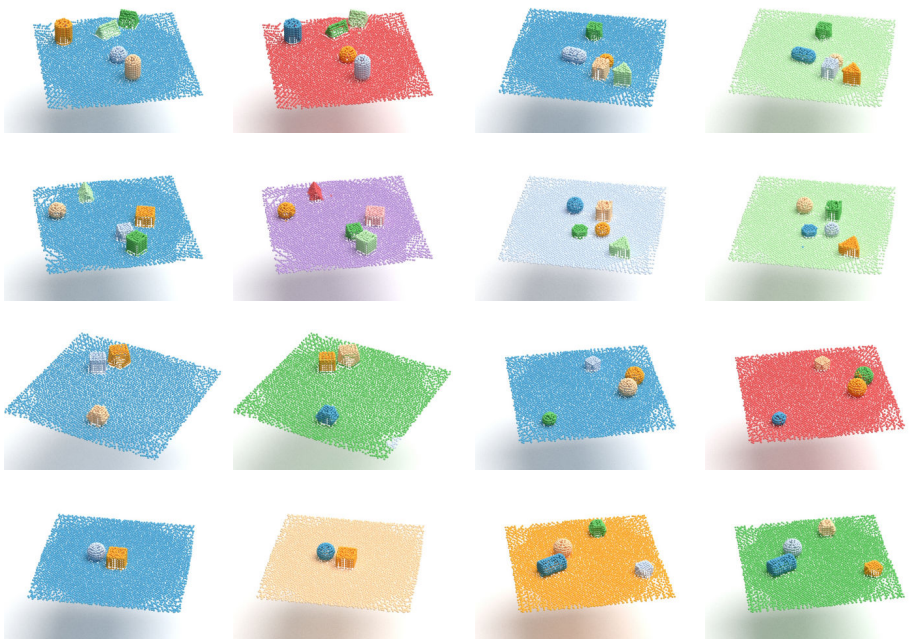
object	figure	min length/max length	min width/max width	min height/max height
Chess piece		0.69/0.86	0.69/0.86	1.43/1.80
Bear		1.1/1.46	0.71/0.95	1.06/1.41
Box		0.98/1.23	0.98/1.23	0.67/0.84
Car		1.17/**1.96**	0.74/1.24	0.70/0.78
Cup		0.87/1.10	0.70/0.87	0.92/1.16
Kettle		0.85/1.13	0.69/0.91	1.01/1.35
Pot		1.06/1.42	0.88/1.17	0.80/1.07
Tissue		1.05/1.31	0.95/1.18	1.16/1.44
Toaster		0.58/0.78	1.01/1.36	0.61/0.81

Table 13: UOT Object pool. Some object meshes are obtained from ai2thor [7] environment.

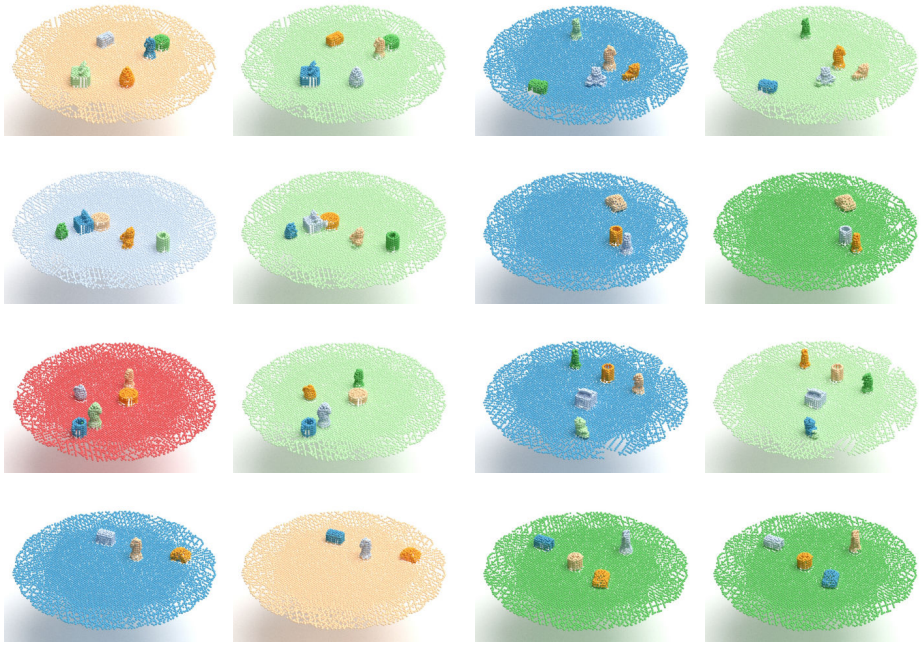
6 More results

We show more segmentation results on UOR in Fig. 5 and UOT in Fig. 6. To provide a more comprehensive inspection, we further display examples of failure segmentation on UOR in Fig. 7, and UOT in Fig. 8, respectively. More segmentation results on scenes with $6 - 12$ objects are shown in Fig. 9 and Fig. 10. In Fig. 11, we show more examples on *Object Matrix* scenes to further demonstrate the scalability of our model.



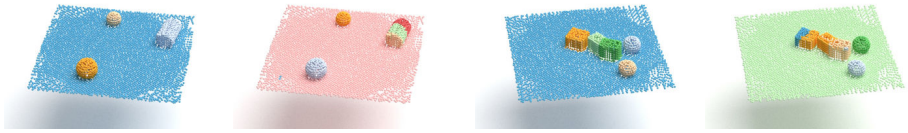
(a) (b) (c) (d)

Fig. 5: UOR segmentation results. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation.



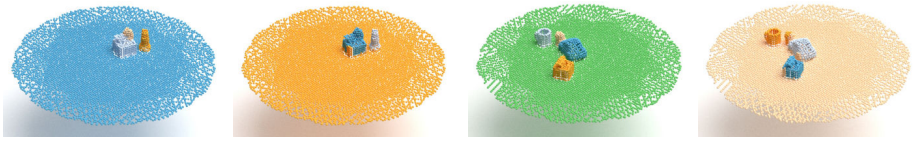
(a) (b) (c) (d)

Fig. 6: UOT segmentation results. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation results.



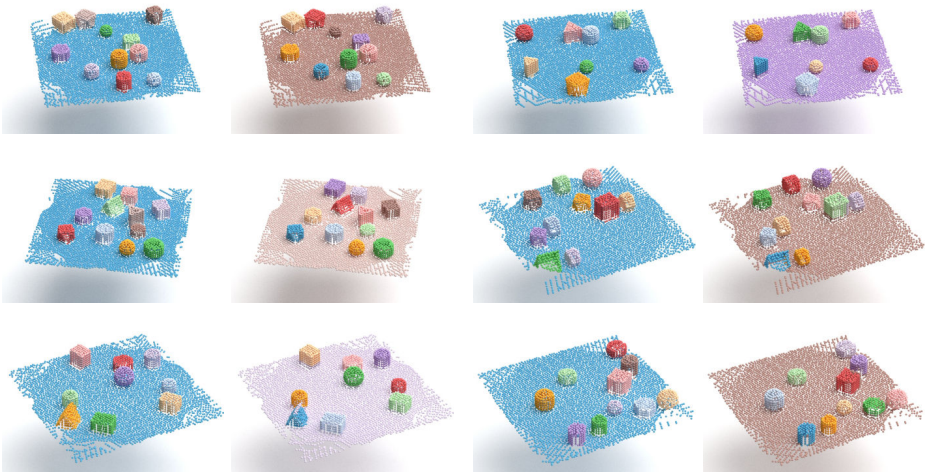
(a) (b) (c) (d)

Fig. 7: UOR failure cases. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation. Failed cases reflect that (1) objects clustered together are more vulnerable to mis-segmentation, and (2) objects with extreme dimensions, e.g. cylinder, are vulnerable to over-segmentations.



(a) (b) (c) (d)

Fig. 8: UOT failure cases. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation results.



(a) (b) (c) (d)

Fig. 9: More results on UOR scenes with 6-12 objects. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation.

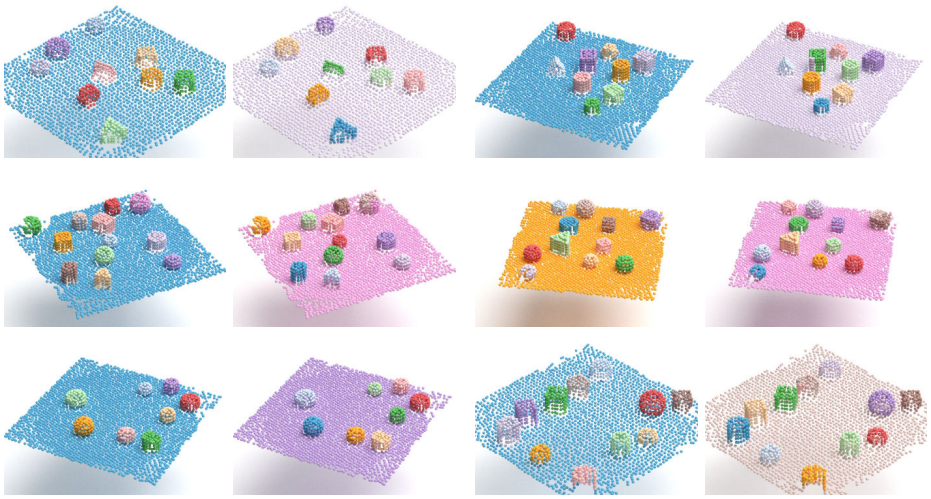
(a) (b) (c) (d)

Fig. 10: More results on UOT scenes with 6-12 objects. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation.

(a) (b) (c) (d)

Fig. 11: More results on UOR and UOT Object Matrix scenes. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation results.

To provide further analysis of our framework, we train our model on the UOR dataset but with different point densities or with noise (but without further hyperparameter tunning). First, we set the point density to $\frac{2}{3}$ of the standard density. Our model achieves **AIR: 0.921**, **SC: 0.827**, and **mSC: 0.836**. Qualitative results are shown in Fig. 12. Then, we set the point density to $\frac{4}{3}$ of that of the standard density Our model achieves **AIR: 0.914**, **SC: 0.831**, and **mSC: 0.833**. Qualitative results are shown in Fig. 13. Finally, we add Gaussian distributed small perturbations to point coordinates. In particular, the standard deviation for the noise is 0.08, which corresponds to objects with the average object radius around 1. Our model achieves **AIR: 0.903**, **SC: 0.817**, and **mSC: 0.825**. Qualitative results are shown in Fig. 14. The three results demonstrate the robustness of our model against input data with varying point density and noises.

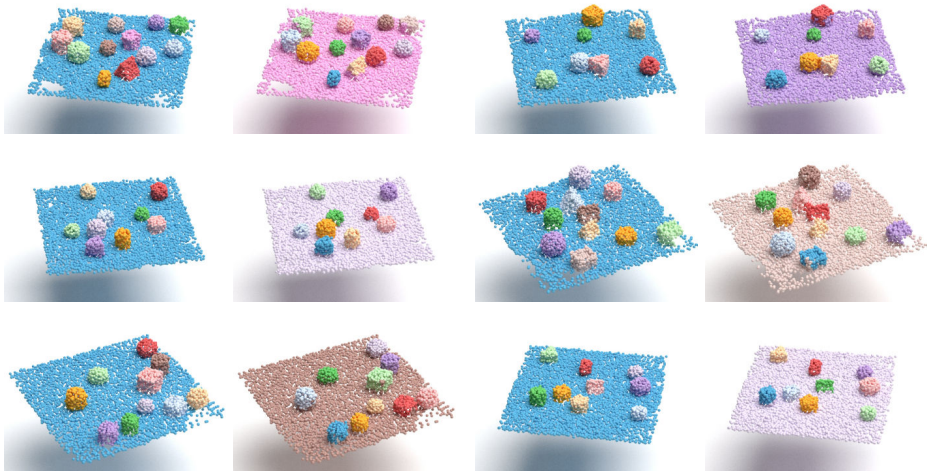


(a) (b) (c) (d)

Fig. 12: Segmentation results on UOR low density variant. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation results.

(a) (b) (c) (d)

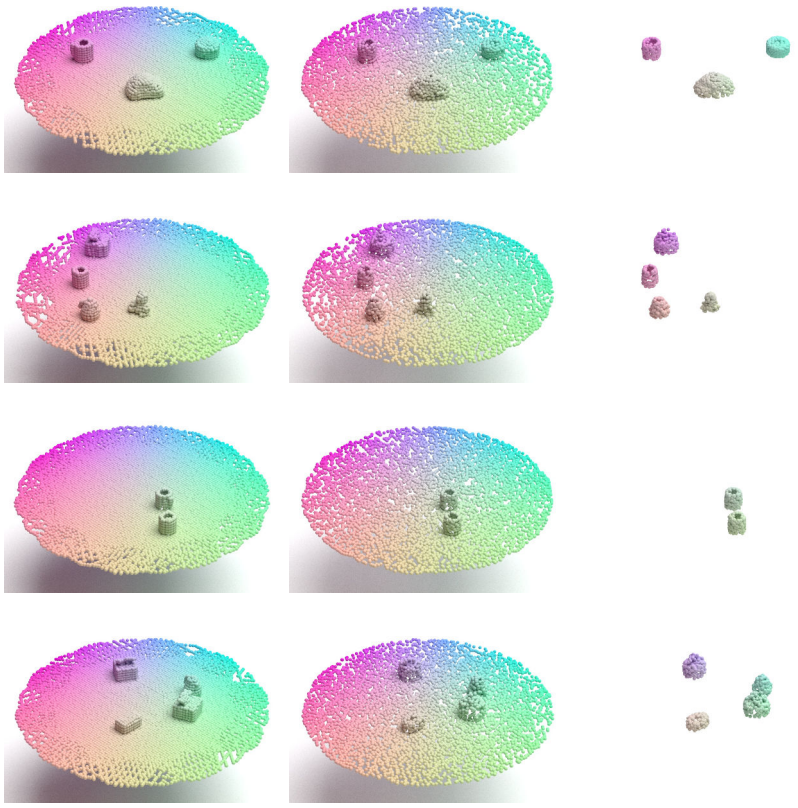
Fig. 13: Segmentation results on UOR high density variant. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation results.



(a) (b) (c) (d)

Fig. 14: Segmentation results on UOR noisy variant. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation results.

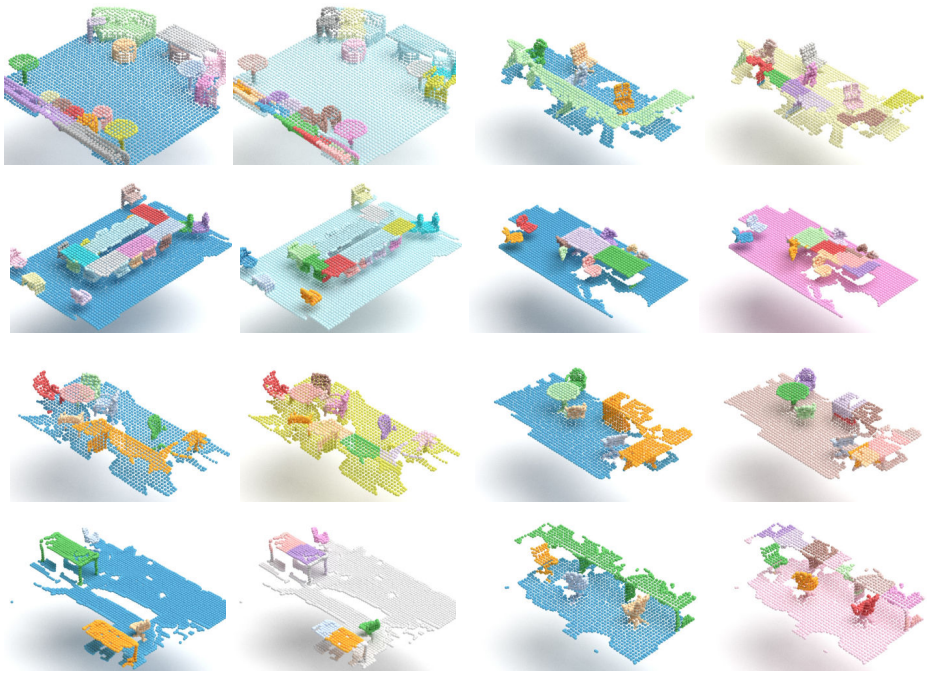
Below we show more we show more reconstruction visualization in individual glimpses.



(a) (b) (c)

Fig. 15: Reconstruction results on UOT dataset. Column (a) is the raw input. Column (b) is the complete reconstruction. Column (c) is the visualization of all foreground glimpses with $z_{pres} > 0.5$.

Below we show more segmentation results on S3DIS dataset.

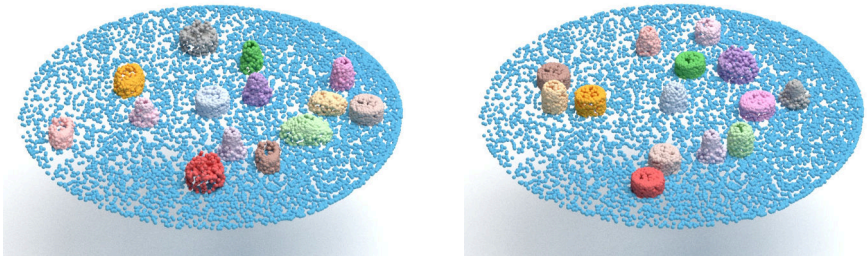


(a) (b) (c) (d)

Fig. 16: S3DIS segmentation results. Column (a) and column (c) are instance labels. Column (b) and column (d) are the corresponding SPAIR3D segmentation results.

Those qualitative results show that chairs are largely segmented with high accuracy except for the ones that are placed under the table with incomplete structure. Long tables are over-segmented into multiple parts as their sizes are larger than the maximum glimpse size. Small round tables are well segmented. Note that the sizes of scenes vary from small offices to large meeting rooms. The spatially invariant property allows our model to perform stably regardless of the scene size.

The SPAIR line of work focuses on learning object-centric representations in their canonical coordinate systems, not capturing the distribution of the scene. Our model thus cannot be used as-is to sample a complete scene satisfying the training set statistics. However, there is no problem in sampling individual objects from **the prior**, following the left branch of Fig. 1 (b)(main paper), and arranging them randomly into a complete scene as shown below.



(a) (b)

Fig. 17: Scene generation example. Each object is sampled from the prior distribution and randomly placed on a scene layout sampled also from the prior.

References

1. Burgess, C., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., Lerchner, A.: Monet: Unsupervised scene decomposition and representation. ArXiv **abs/1901.11390** (01 2019), https://arxiv.org/abs/1901.11390
2. Crawford, E., Pineau, J.: Spatially invariant unsupervised object detection with convolutional neural networks. AAAI **33**, 3412–3420 (07 2019). https://doi.org/10.1609/aaai.v33i01.33013412
3. Greff, K., Kaufman, R.L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M.M., Lerchner, A.: Multi-object representation learning with iterative variational inference. In: ICML (2019)
4. Greff, K., van Steenkiste, S., Schmidhuber, J.: Neural expectation maximization. In: NeurIPS. p. 6694–6704. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
5. Jiang, L., Zhao, H., Shi, S., Liu, S., Fu, C.W., Jia, J.: Pointgroup: Dual-set point grouping for 3d instance segmentation. CVPR (2020)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., Farhadi, A.: AI2-THOR: An Interactive 3D Environment for Visual AI. arXiv (2017)
8. Li, N., Eastwood, C., Fisher, R.: Learning object-centric representations of multi-object scenes from multiple views. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 5656–5666. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/3d9dabe52805a1ea21864b09f3397593-Paper.pdf
9. Shi, W., Rajkumar, R.: Point-gnn: Graph neural network for 3d object detection in a point cloud. In: CVPR. pp. 1708–1716 (2020)
10. Wu, W., Qi, Z., Li, F.: Pointconv: Deep convolutional networks on 3d point clouds. In: CVPR. pp. 9613–9622 (06 2019). https://doi.org/10.1109/CVPR.2019.00985