A APPENDIX

A.1 MINIMUM VOLUME PRINCIPLE IN DETAIL

The proposed approach for recovering the pose of an object is described in Algorithm 1. A visualisation of how the estimated pose evolves as the volume of the AABB that contains the object decreases is depicted in Figure 5. An advantage of the proposed minimum volume principle is that the recovered pose will align one of its coordinate planes with the symmetry plane of the object if it exists.

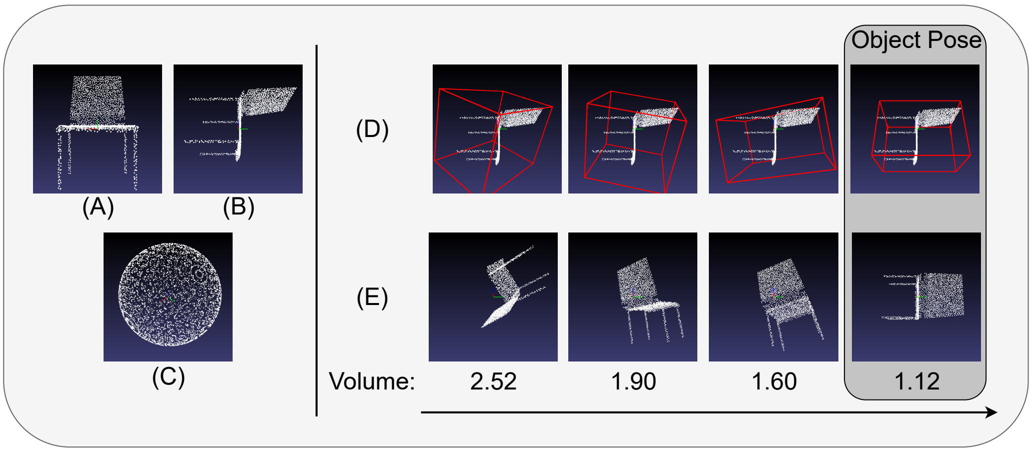

Figure 5: We demonstrate the point cloud of a chair (A) in arbitrary pose (B) and how its pose is recovered using the minimum volume principle. The selection matrices $\mathbf{R}^s$ are illustrated in (C) by rotating a point at $[1, 1, 1]$ with all the selection matrices. (D) and (E) depict how the estimated object orientation represented by a bounding box and the object viewed with respect to its reference pose evolve with respect to the volume of the AABB.

Algorithm 1: Minimum Volume Principle

Input: points $\mathbf{P} \in \mathbb{R}^{N\times 3}$ representing the object, and selection rotation matrices $\mathbf{R}^s \in \mathbb{R}^{N_s\times 3\times 3}$.
Output: translation and rotation parameters $\{\mathbf{T}, \mathbf{R}\}$ that map $\mathbf{P}$ to their cannonical coordinates

```
$\beta \leftarrow 0.01$    ▷ Tolerance hyperparameter
$\hat{\mathbf{P}} = (\mathbf{R}^s)^{-1}\mathbf{P}$    ▷ Transform points into coodinates with orientations $\mathbf{R}^s$
$V = \textsc{ComputeVolume}(\hat{\mathbf{P}})$    ▷ Compute volume of AABBs that contain $\hat{\mathbf{P}}$
$V_{\text{min}} = \min(V)$    ▷ Find smallest volume
$\hat{\mathbf{R}}^s = \{\mathbf{R}^s_i | V_i \in [V_{\text{min}}, (1+\beta)V_{\text{min}}]\}$    ▷ Select similar sized AABBs to minimum
$d_{\text{SO}(3)} = \arccos\left(\frac{\text{tr}(\hat{\mathbf{R}}^s)-1}{2}\right)$    ▷ Compute geodesic distance to the world frame
$\mathbf{R} = \hat{\mathbf{R}}^s_j$ with $j = \text{argmin}(d_{\text{SO}(3)})$    ▷ Find AABB closest to the world frame
$\mathbf{T} = \frac{1}{N}\sum_{i\in\{1,\ldots,N\}} \mathbf{p}_i$ with $\mathbf{p}_i \in \mathbf{P}$    ▷ Compute centre of mass for points
```

A.2 TRAINING AND IMPLEMENTATION DETAILS

Training takes about 35 hours for the YCB moving-object dataset and 20 hours for the YCB static dataset on a single NVIDIA Titan RTX GPU. The video model and the static model are trained for 100k and 100.5k iterations, each with a batch size of 2 and 8, respectively, selected according to the available GPU memory. The IC-SBP ablation has a batch size of 8 for the video experiments as it's a static image-based approach. Video length is 5 frames randomly cropped from the full video for training and we use the full video length (typically 15 frames or longer) for testing. The model is thus able to process videos that are much longer than the videos seen during training. The YCB moving-object dataset has 4000 videos for training and 500 videos for testing. The YCB static dataset has 12000 scenes for training and 1500 scenes for testing. We summarise all the hyper-parameters in Table 3. In the ablation study we use slot attention with a background slot as introduced in Yu et al. (2021). For video data, slots from the last time step are used to initialise slot attention at the current time step. For both datasets, OBPOSE estimates poses with all points below the ground surface being neglected, as we find that NeRF can mistakenly predict the occupancy of the points below the ground surface whose x-y coordinates are within the object's convex hull. This (erroneous) generalisation follows from the fact that this part of the scene receives no training signal, due to the ray marching used in NeRFs. For the training on the CLEVR dataset, we find that using a L2 loss rather than a mixture of Gaussian loss for the eq. (13) can improve the robustness of the training, i.e.:

$$\mathcal{L}_{\text{att}} = \sum_{k=1}^{K} (\mathcal{N}(\mathbf{x}|\mathbf{c}_k, \sigma_{\text{std}}^2)/\mathcal{N}(0|0, \sigma_{\text{std}}^2) - \mathbf{m}_k)^2 + \sum_{k=1}^{K} (\mathbf{m}_k - \hat{\sigma}_k^{\text{surface}}/\sigma_{\text{max}})^2 \tag{16}$$

A.3 NETWORK STRUCTURES

The KPConv encoder backbone consists of an input block and three consecutive encoding blocks with radiuses of $[0.025, 0.05, 0.1, 0.2]$ meters for the KPConv layers. Each block has three sub-blocks which each has a structure of *Conv1d-Group norm(Wu & He, 2018)-ReLU-KPConv-Conv1d-Group norm-ReLU* with a skip connection being built between the output of the first ReLU layer and the output. The skip connection takes the output of the ReLU and feeds it into a Conv1d layer followed by a Group norm. The input block consists of an input layer of *KPConv-Group norm-ReLU* and a single sub-block as described above. For the input block and all the sub-blocks, the number of channels are $[16, 16, 32, 32, 32, 64, 64, 64, 64, 64, 64]$ respectively and the strides are $[8, 1, 4, 1, 1, 4, 1, 1, 4, 1, 1]$. The KPConv-aggregate layer has a channel number of 64 and a radius of 1 meter. Two trilinear interpolation layers are used to up-sample the embedding and predict embedding of 64 channels and 32 channels respectively. A *Conv1d-Group norm-ReLU* encoding is used in both the input and the skip connection between the input and the output of the interpolation layer. A final output layer of structure *Conv1d-Group norm-LeakyReLU(0.1)-Conv1d* is appended to the output embedding with channel numbers of 32 and 16 for the two Conv1d layers. The *where* encoder and the *what* encoder share the same structure of the backbone encoder and have channel numbers of $[16, 16, 32, 32, 32, 64, 64, 64, 64, 64, 64]$ and strides of $[1, 1, 4, 1, 1, 4, 1, 1, 4, 1, 1]$. A final output layer of structure *Linear-Group norm-LeakyReLU(0.1)-Linear* is used to predict the final embedding. Both linear layers have 64 output channels. The RNNs used to parameterise the posterior and the prior distribution of the *where* and the *what* latent embedding has a dimension of 32 for the hidden state followed by an MLP of size $[32, 32, 64]$ with the output 64 channels being further divided into the mean and the standard deviation of the Gaussian. The $\mathbf{z}^{where}$ is decoded into the $\Delta\mathbf{T}$ using an MLP of sizes $[32, 256, 3]$. We use the default NeRF decoder open-sourced by Niemeyer & Geiger (2021).

Table 3: Hyperparameters of OBPOSE.

Parameter	Description	Value
σ_{std}	Standard deviation of colour dist.	0.1
σ_{max}	Maximum density	10/1 for YCB/CLEVR
s	Bounding box size	0.4/0.2 for YCB/CLEVR
N_{thresh}	Number of points to use the pose of last time step	10
T_{max}	Maximally allowed $\Delta\mathbf{T}$	0.1
$\mathbf{S}$	Number of sparse voxels along each dimension	24
δ	Noise added to depths d_{surface}	0.01/0.7 for YCB/CLEVR
β	tolerance factor to select minimum volume bounding box	0.1/0.01 for video/static images
K	Number of slots for IC-SBP	4
d_z^{what}	Dimensionality of $\mathbf{z}^{what}$	32/256 for YCB/CLEVR
d_z^{where}	Dimensionality of $\mathbf{z}^{where}$	32

A.4 SEGMENTATION RESULTS ON YCB STATIC

We illustrate additional segmentation results on YCB static dataset in Figure 6.

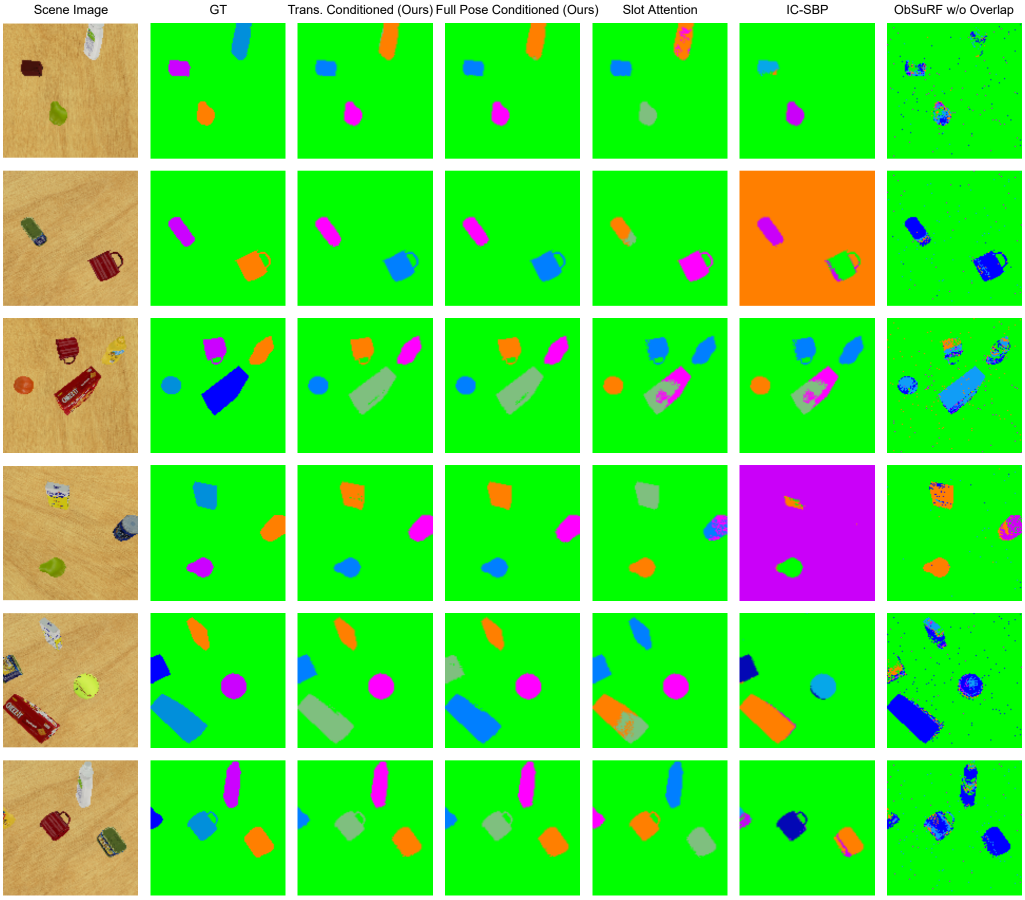

Figure 6: Qualitative results on YCB static dataset.

A.5 RENDERING RESULTS ON YCB STATIC

We illustrate additional NeRF rendering results on YCB static dataset in Figure 7.

A.6 RESULTS ON CLEVR-3D

We show qualitative results on CLEVR-3D dataset in Figure 8. We find that OBPOSE can segment part of the shadows of the objects as part of the objects because the shadows are caused by the existence of the objects. The shape of the objects can be recovered even OBPOSE is trained without multi-view supervisions.

A.7 COMPARISON BETWEEN VIDEO INPUTS AND SINGLE FRAME INPUTS

We show OBPOSE's performence taking video inputs and single frame inputs in Table 4 and Figure 9 respectively. We find that OBPOSE's performance is only slightly worse with single frame inputs, but the ability of tracking objects is lost in that input setting.

A.8 MORE QUALITATIVE RESULTS

We illustrate additional segmentation results on CLEVR-3D and MultiShapeNet datasets in Figure 10.

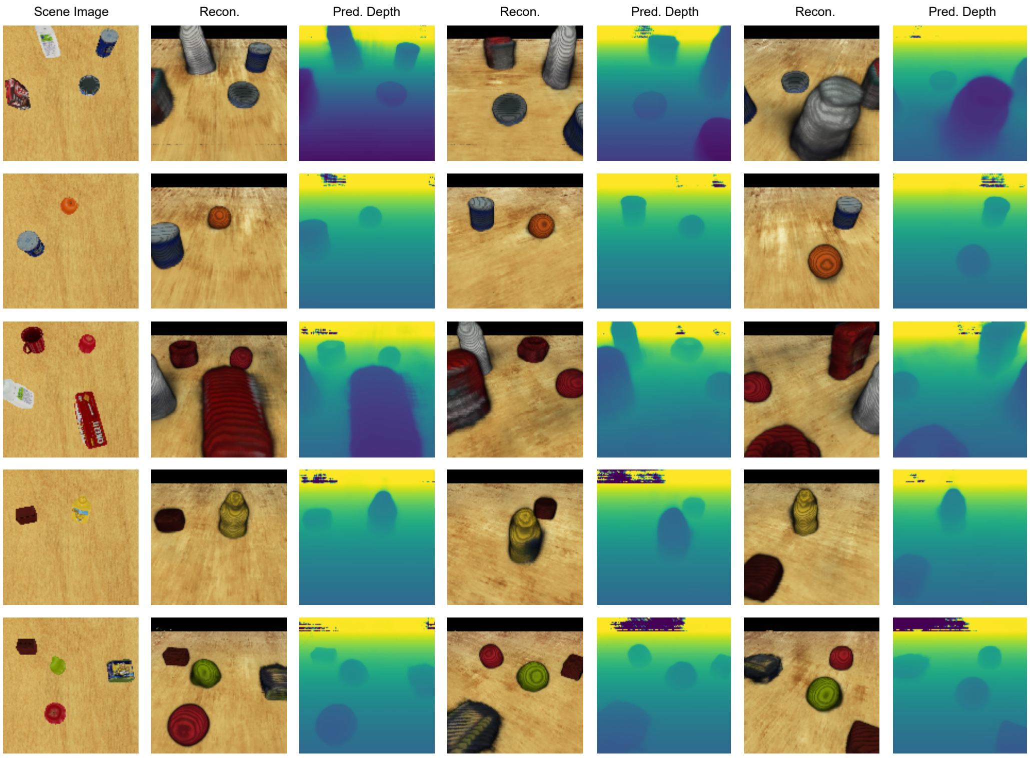

Figure 7: NeRF rendering results on YCB static dataset.

Table 4: The segmentation results on the YCB moving-object dataset evaluated on video inputs and single-frame inputs. The results are rounded to two decimal places.

	mIoU-BG↑	ARI-FG↑	MSC-FG↑
OBPOSE + VIDEO INPUTS	1.00 ± 0.00	0.96 ± 0.00	0.97 ± 0.00
OBPOSE + SINGLE FRAME INPUTS	1.00 ± 0.00	0.94 ± 0.00	0.97 ± 0.00

A.9 POSE ESTIMATION FOR VARIANCE REDUCTION ENCODING

We show how the variance of the object appearance can be reduced by leveraging the shape-based estimated object location and orientation for the encoding of the objects in the fig. 11.

A.10 LIMITATIONS AND FAILURE MODE

The present work advances the state-of-the-art in unsupervised 3D-scene segmentation and interpretation. While OBPOSE is important for advancing research, its impact outside of the machine learning community is low as current methods can not yet deal effectively with real images. This highlights the current limitations of the proposed model. In the future, we will investigate how to build a model that can explain complex real-world observations (e.g. including shadows). OBPOSE will then be ready for real-world robotics applications as a vision backbone.

An important failure mode of OBPOSE is that our model can not predict meaningful object orientations when the object is severely occluded. In this case, we thus do not condition the encoding on the object orientation but only on the object location. Also, we found that the wooden box in the YCB static dataset is segmented as part of the background. We hypothesise that this is due to it having a similar texture to the wooden table (background). However, the wooden box is segmented correctly in the YCB video dataset, which would appear to suggest that the temporal structure of video can improve object segmentations.

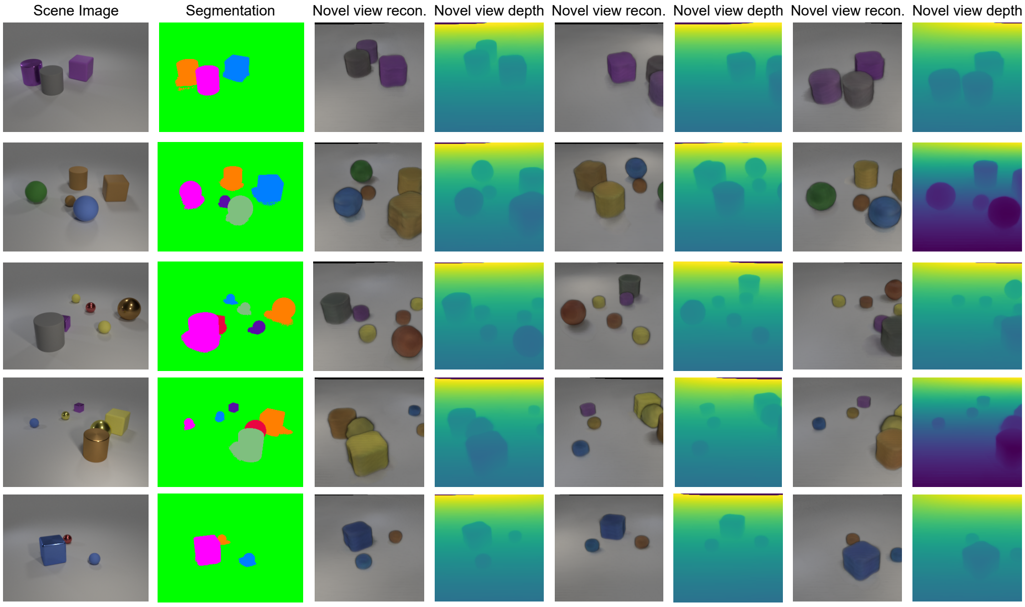

Figure 8: Qualitative results on CLEVR-3D dataset.

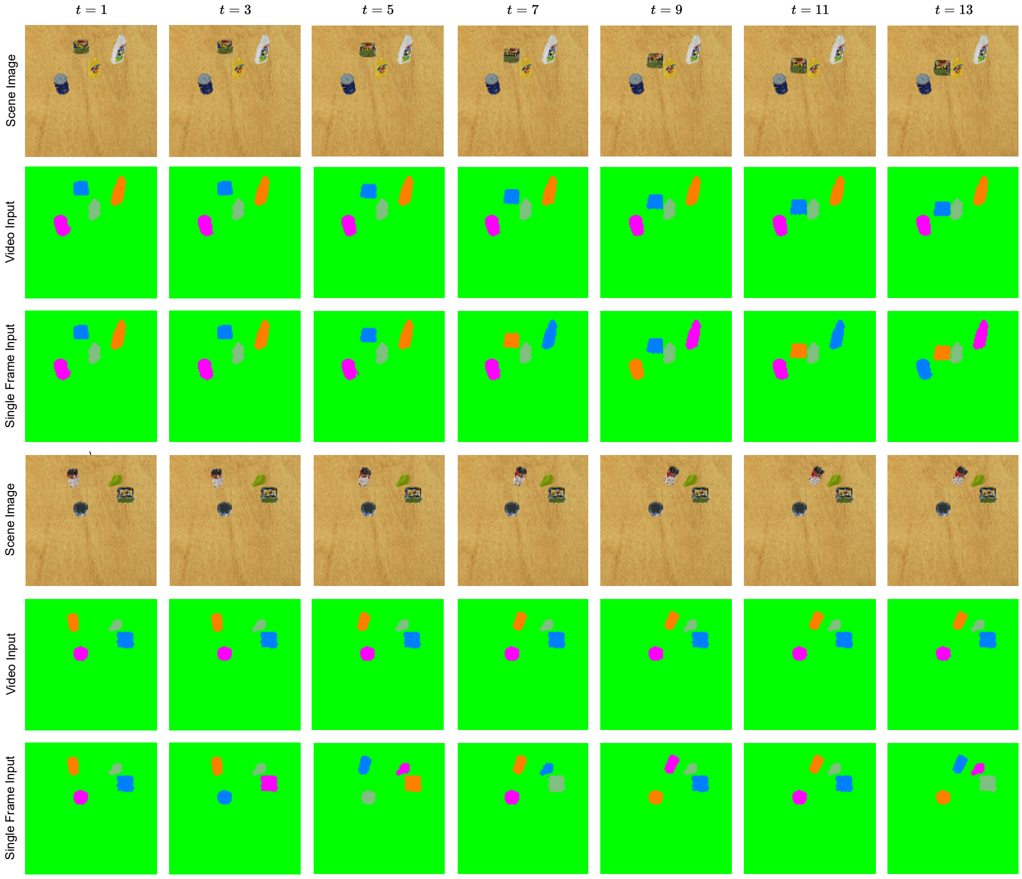

Figure 9: Qualitative results on YCB moving-object dataset. Performance is evaluated on video inputs and single frame inputs respectively. While segmentation looks similarly good in both cases, note that the single frame input results lose the ability to track objects, as indicated by the occasional change in object mask colourbetween frames.

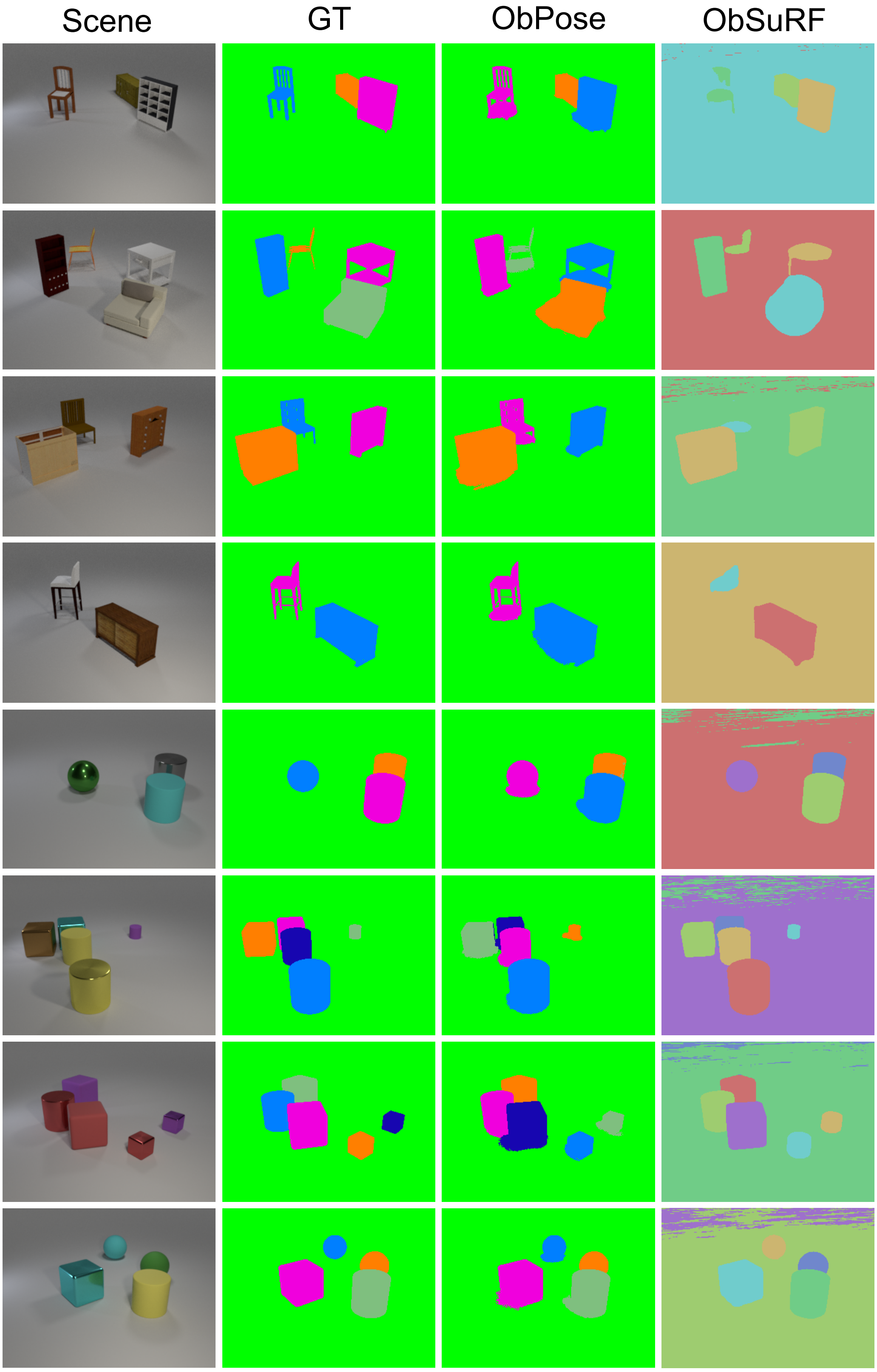

Figure 10: Qualitative results on CLEVR-3D and MultiShapeNet. We find that ObPose can successfully identify part of the shadows of the objects. This is contradictory to the ground-truth labels where all object shadows are labelled as background.

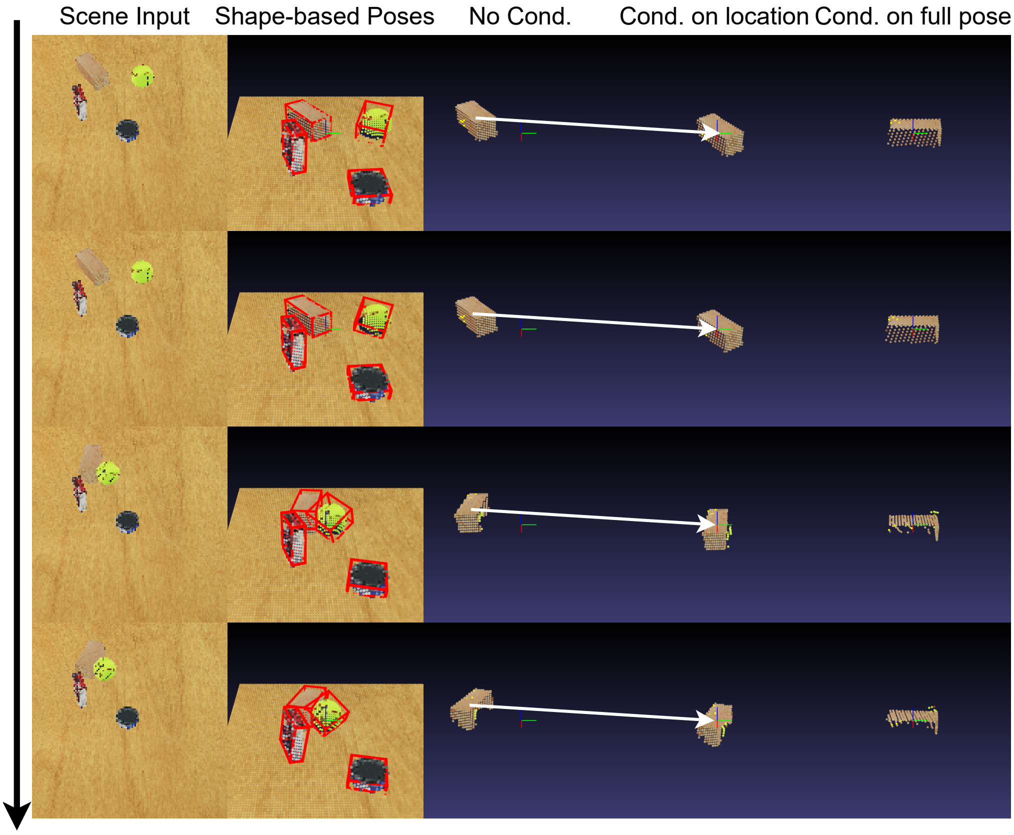

Figure 11: Visualisation of the recovered bounding boxes estimated from the object shape. We additionally illustrate the object (wooden box) appearance viewed by (a) being not conditioned on any location information (b) being conditioned on the shape-based estimated object location and (c) being conditioned on the shape-based estimated object location and orientation. We show that the variance of the object appearance is reduced leveraging the pose information.