7. Supplementary Material

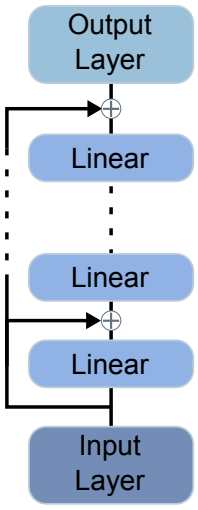

Figure 7. **MLP Block with Skip-Connections.**

7.1. Synthetic Training Data Generation

For the generation we proceed as following: We pick a random number of objects, sample them in a scene and render 20 PBR images from random camera positions focusing on a random object in the scene for each camera position. The scenes vary in objects placed (i) randomly on the ground, (ii) flying in space, or (iii) in an upright standing position. We keep generating scenes until all objects have been seen (*i.e.* at least 40% visibility) at least 64 times in at least 8 different scenes. Afterwards we convert the scene-centric into a highly efficient object-centric format by mapping the object id to all images where they occur, giving us at least 64 random views for each object in each scene type. See Fig. 8 for some example images from the training set.

7.2. Losses

Encoder loss. To enforce the encoder to predict towards ground truth surface points we define a simple regression loss that minimizes the Euclidean distance between the estimated surface point $s_i = q_i - \widetilde{\mathcal{U}}(q_i) \times \frac{d\mathcal{U}(q_i)}{dq_i}$ with $\widetilde{\mathcal{U}}(q_i) := \mathcal{U}\left(\widehat{f}_i^{\,\mathcal{Q}}(q_i)\right)$ and the ground truth surface point $\bar{s}_i$:

$$\mathrm{L}_{\chi} = \frac{\sum_{i=1}^{M} \|s_i - \bar{s}_i\|}{M}, \tag{5}$$

with $M = |\chi|$. We do not directly enforce any loss on the NeMO features $\widehat{\mathcal{F}}^{\mathcal{Q}}$ so that they can be freely learned through backpropagation through the decoder. During training, after Ψ has predicted χ, we randomly rotate, translate and scale the NeMO points s_i by a random transformation $\mathbf{T}$ to teach the decoder to learn a coordinate system that is independent of the anchor image I_{A}.

Dense Prediction Losses. We define additional losses on the dense predictions of the decoder Θ. For the modal and amodal segmentation losses $\mathrm{L}_{\text{modal}}$ and $\mathrm{L}_{\text{amodal}}$ we use an equally weighted dice- and binary cross-entropy loss. The pointmap is learned using a confidence weighted L1 loss: We define the set of valid pixels in I_{q} belonging to the object as $\mathcal{D}_{\text{Obj}}$. As in [60] we define a regression loss between the estimated 2D-3D correspondences X and the ground truth mapping $\bar{X}$:

$$\mathrm{L}_{\text{2D3D}} = \frac{\sum_{i \in \mathcal{D}_{\text{Obj}}} \widetilde{C}_i \|X_i - \bar{X}_i\|}{|\,\mathcal{D}_{\text{Obj}}\,|} \tag{6}$$

where $\widetilde{C}_i = 1 + \exp C_i$ makes sure that a confidence of zero does not lead to diminishing gradients. Note that during training we transform the ground truth correspondences $\bar{X}$ by the same transformation $\mathbf{T}$ as applied to the points s_i in χ. Different to [60], we explicitly enforce certainty on pixels belonging to the object and uncertainty on pixels that do not belong to the object:

$$\begin{aligned} \mathrm{L}_{\text{certain}} &= \frac{\sum_{i \in \mathcal{D}_{\text{Obj}}} \Big(1 - \tanh\big(\exp(C_i)\big)\Big)}{|\mathcal{D}_{\text{Obj}}|} \\ \mathrm{L}_{\text{uncertain}} &= \frac{\sum_{i \in \mathcal{D}_{\text{Bg}}} \Big(\tanh\big(\exp(C_i)\big)\Big)}{|\mathcal{D}_{\text{Bg}}|} \end{aligned} \tag{7}$$

where $\mathcal{D}_{\text{Bg}}$ are all pixels not belonging to the object. Based on the introduced losses, we define the total training loss as:

$$\begin{aligned} \mathrm{L}_{\text{total}} &= \alpha(\mathrm{L}_{\chi} + \mathrm{L}_{\text{2D3D}}) \\ &\quad + \beta(\mathrm{L}_{\text{certain}} + \mathrm{L}_{\text{uncertain}} + \mathrm{L}_{\text{modal}} + \mathrm{L}_{\text{amodal}}), \end{aligned} \tag{8}$$

where α, β are hyper-parameters. During training we set $\alpha = 1.0, \beta = 0.2$.

7.3. Training Details

Sample Structure. We define each sample in a batch as ten randomly picked images from the training set showing the same object and 1500 sampled points in the volume $[-1, 1]^3$. Five of the images are used as template images for the encoder to generate a NeMO, while all ten images are used as input to the decoder. The total loss as defined in Eq. (8) includes the encoder as well as multiple dense decoder losses, training the encoder and decoder jointly. While the five images used for the NeMO generation always show the same object with small variations in the bounding box, the other five images have a 30% chance of being a completely random crop from the scene. This allows the

decoder to learn to ignore images in which the object is not shown while also improving the decoder's capability to find the object in cluttered scenes.

Data Augmentation. In $2/3$ of the samples we use 1500 uniform sampled 3D points in the volume $[-1, 1]^3$ as input $\mathcal{Q}$ while in $1/3$ of the samples we use 1500 points sampled close to the surface of the CAD model of the object. Due to the self-attention mechanism in the encoder, this allows us to train the model to use the CAD model information if available, while also being able to work without it. As pixel-level data augmentation to all templates in a sample, we apply:

- 30% chance of color jitter with 0.5 brightness, 0.5 contrast, 0.5 saturation and 0.5 hue, and
- 10% chance of grayscale,

On template images directly we apply:

- 50% chance of random affine transformation with up to 5° rotation, 0.1 translation and 0.1 scaling, and
- 50% chance of Gaussian noise with standard deviation between 0 and 0.05.

Initial NeMO Space. To define the initial coordinate system based on the anchor image I_{A}, we use the mean of the ground truth surface points as the center of the NeMO space and the orientation of the anchor image as it is in the camera frame in OpenCV notation. The scale of the NeMO point cloud is defined such that the anchor image's surface points take $1/3$ of the volume $[-1, 1]^3$.

7.4. Object Alignment

As the BOP challenge compares the predicted object pose against the ground truth pose, we need to align the coordinate system of our NeMO with the ground truth object pose to be able to evaluate our approach. Note that this is only required for evaluation and generally not a necessity of our method. As we are given the object-to-camera transformation both in the model-based, where we render our templates, and in the model-free setting, where we are given the ground truth pose, we can align our coordinate system by optimizing the scale, rotation and center-offset. We use a simple gradient descent optimization that minimizes the error between the predicted and ground truth pose of the $\text{k} = 5$ best rotation predictions of the template images. See Algorithm 1 for the pseudo code of the alignment algorithm.

7.5. From Dense Predictions to Detection, Segmentation and 6DoF Pose Estimation

Since our decoder produces dense predictions, we need additional steps to get a 6DoF pose estimation and amodal bounding box. These predictions are run in parallel using multiple decoder and different NeMOs. To predict the segmentations and amodal bounding boxes of a scene, we patchify the scene image with two patch sizes (smallest image side and half the smallest image side) with a stride of $1/3$ of the patch size. We accumulate the patch predictions and look for clusters of predictions larger 0 to differentiate between object instances. The amodal bounding box is retrieved by laying a box around a amodal segmentation cluster. We ignore small clusters and clusters touching the image sides. To predict the 6DoF pose estimation, we crop the region from a bounding box and all pixels with a minimum confidence value of 0.1 in the 2D-3D correspondences. We then compute the pose by solving the PnP problem using OpenCV's `solvePnPRansac` function with `iterationsCount 500`, `reprojectionError 6` and `min inlier ratio` of 30%.

Algorithm 1 NeMO Alignment Algorithm

Require: Ground truth poses $T_i^{\text{gt}} \in SE(3)$, estimated poses $T_i^{\text{est}} \in SE(3)$, max iterations N, learning rate η, Huber loss threshold δ, number of best poses k
Ensure: Optimal scale s, rotation R, translation t that aligns T^{est} to T^{gt}

```
Select k best estimates with lowest initial rotation error
Initialize scale s based on average translation norms
Initialize rotation as axis-angle vector r ← 0, translation t ← 0
for i = 1 to N do
    R ← AxisAngleToRotation(r)
    Construct correction transform T^corr = [R t; 0 1]
    Apply correction: T̂_i ← T_i^est · T^corr
    Compute predicted R̂_i and scaled t̂_i
    Compute rotation error θ_i = ∠(R̂_i, R_i^gt)
    Compute translation error ε_i = ‖t̂_i − t_i^gt‖
    Compute robust loss using Huber: L = Huber(θ) + Huber(ε/10)
    Update s, r, t via gradient descent with Adam optimizer
    if L < L_best then
        Save current s, R, t as best parameters
    end if
end for
return Optimized s, R, and t
```

7.6. Additional NeMO Analyses

Similarity between predicted and ground truth object surface. We show the close resemblance of our dense point cloud by measuring the Chamfer distance to the ground truth point cloud. The prediction is based on two different point cloud sampling methods as input to our NeMO. Both methods achieve strong results in Tab. 7 across the three datasets with slightly better performance, when the input point cloud is sampled based on the actual CAD model for a given object.

Sampling Method	Chamfer Distance ↓		
	YCB-V	HOPEv2	HANDAL
Random Sampled	0.0055	0.0029	0.009
CAD Surface	0.0036	0.0019	0.0022

Table 7. **Chamfer distance between ground truth point cloud and dense point cloud prediction based on two different NeMO input sample point clouds.** First, randomly sampled points. Seconds, points sampled on the CAD surface, which requires the model of a given object. We observe already small errors only in case of randomly sampled point clouds as input to our NeMO representation. We further improve this distance when we sample points based on the provided CAD model.

ICP Method	Point Sampling	AP ↑	AP_{MSPD} ↑	AP_{MSSD} ↑	Time/Image (s) ↓
–	Random	0.378	0.370	0.385	0.336
–	CAD Surface	0.383	0.420	0.347	0.531
Pointmap Surface	Random	0.475	0.394	0.557	1.720
Pointmap Surface	CAD Surface	0.507	0.447	0.567	1.479
CAD	Random	0.671	0.639	0.703	2.049
CAD	CAD Surface	0.626	0.594	0.658	1.607

Table 8. **Precision with different point sampling and ICP.** We report the AP on YCB-V with ground truth detections without refinement.

CAD-Model Conditioned Sampling. We compare the performance of our network using randomly sampled $\mathcal{Q}$ and points sampled from the surface of a CAD model. Additionally, we compare different ICP-based refinement strategies: (i) no ICP, (ii) using the pointmap prediction as source point cloud for the ICP, and (iii) using CAD model surface points as source point cloud for the ICP. The results are shown in Tab. 8. We show that our design choice to use a continuous point cloud as input to the encoder can leverage the information from the CAD model to improve the performance of our method. We also show that we can use the pointmap prediction for ICP refinement, although no depth information is given during the encoding step, increasing the average precision compared to purely RGB based (no ICP) pose estimation. An example of estimated object surfaces can be seen in Fig. 10. This experiment demonstrates the versatility of our network, allowing us to adapt to different scenarios.

Method	T-LESS	TUD-L	YCB-V
Co-op (F3DT2D, Coarse, RGBD)[41]	0.620	0.841	0.808
FreeZeV2.1[8]	0.751	0.991	0.905
FreeZeV2.2[8]	0.777	0.988	0.911
FRTPose-WAPP (Default Detection)	**0.826**	0.978	**0.918**
Ours (CNOS Detection)	0.291	0.560	0.592
Ours (NeMO detection)	0.144	0.756	0.567

Table 9. **Model-Based 6D Detection of Unseen Objects with Refinement.** We report Average Precision (AP) on 3 BOP datasets and compare against other methods published on the public Model-Based Unseen Object 6D Detection leaderboard. [3]. We used a simple ICP as refinement step. † indicates unpublished methods.

NeMO Generation	Chamfer Distance ↓		
	YCB-V	HOPEv2	HANDAL
Fused	0.0080	0.0034	0.0088
All Templates	0.0081	0.0034	0.0084

Table 10. **Chamfer distance between ground truth point cloud and dense point cloud prediction based on two different NeMOs.** First, a fused NeMO based on two respective NeMOs generated through 4 images each but each with the same anchor image. Second, predicted dense point cloud based on all 7 template views. This shows the versatility of our NeMO as the point cloud resemblance and error stays constant.

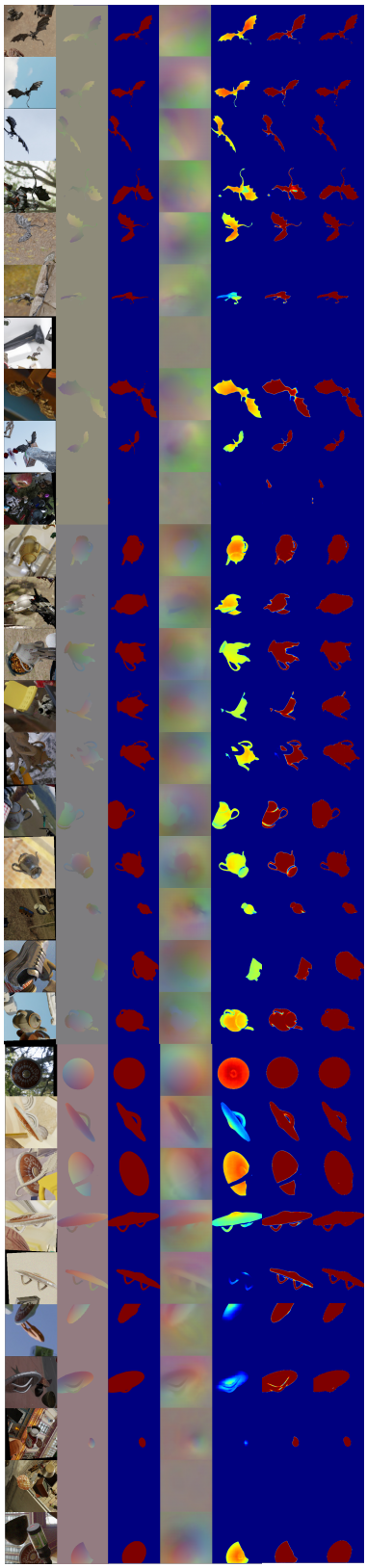

Figure 8. **Training Images.** From left to right, respectively: RGB input, ground truth 2D3D mapping, ground truth amodal segmentation, estimated 2D-3D mapping and estimated confidence, estimated modal segmentation and estimated amodal segmentation. Shown are three different training samples, each sample consists of 10 images. The first 5 images are used to generate a NeMO, all 10 images are used for dense predictions. Note that in the last 5 templates of a sample, the object crop is stronger augmented and sometimes the object is not visible at all.

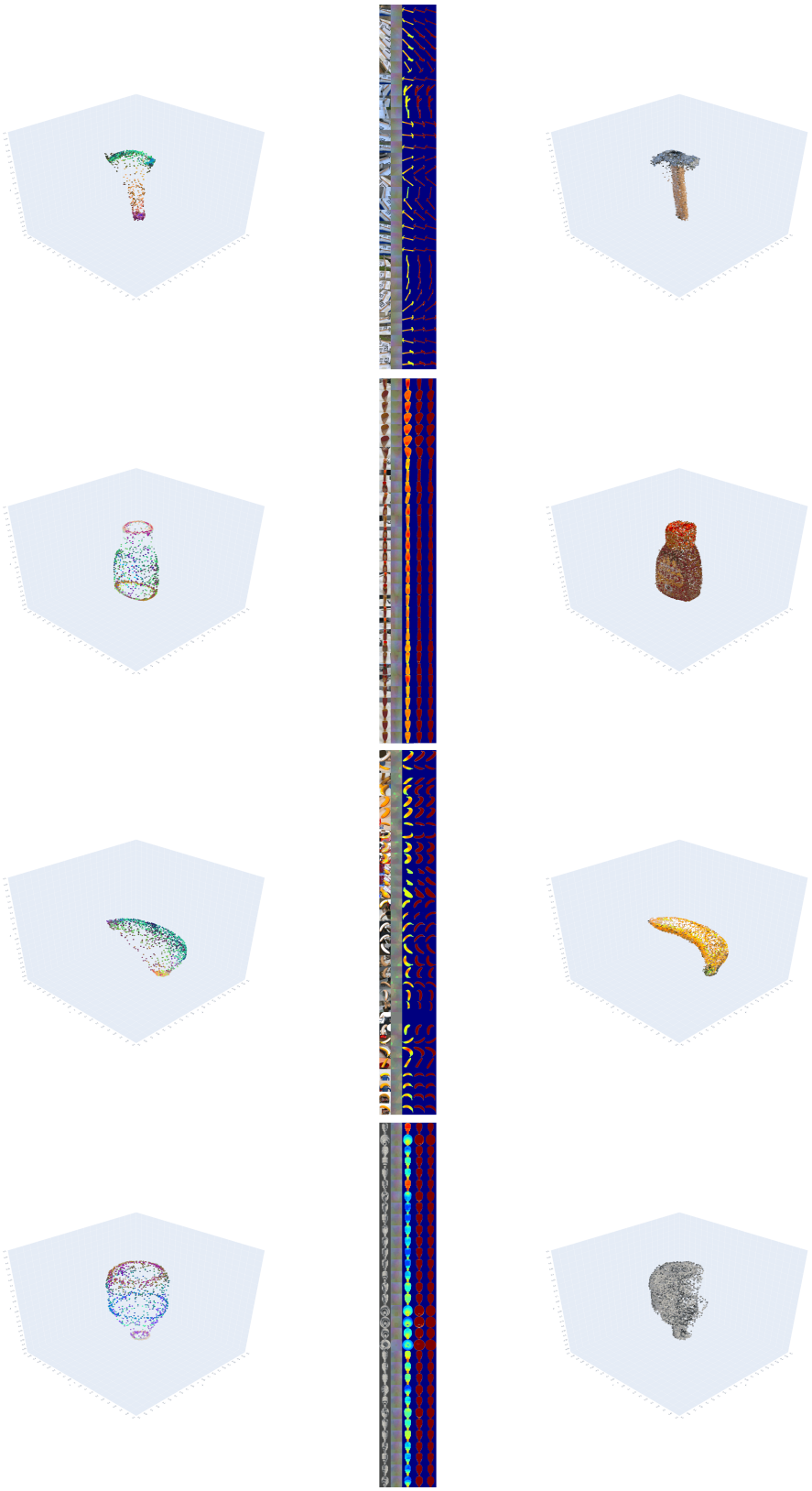

Figure 9. **NeMOs from benchmark objects.** We show the NeMOs (left, with PCA feature visualization), dense predictions (middle) and object surface reconstruction (right) from objects selected from different datasets. From top to bottom, the datasets are: HANDAL, HOPEv2, YCB-V and T-LESS. We use 32 template images to generate the NeMOs and use the same 32 images to show the dense prediction results. Note that we show real template images for HANDAL, HOPEv2 and YCB-V and synthetic template images for the T-LESS object. All NeMOs have been aligned with the ground truth pose as described in Sec. 7.4.

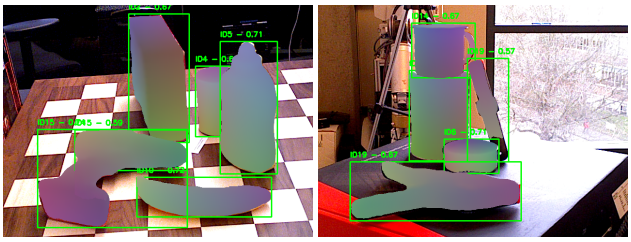

Figure 10. **Pointmap predictions** on the YCB-V dataset. We only show estimations with confidences > 0.1.

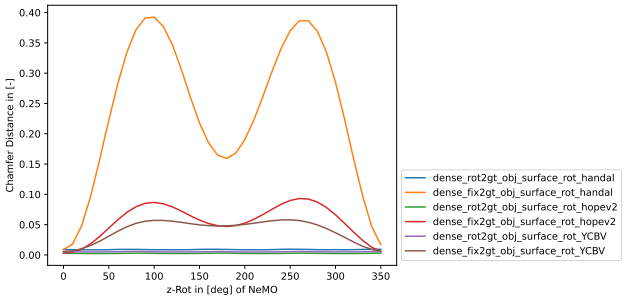

Figure 11. **Chamfer distance between dense predicted and rotated ground truth point cloud** averaged across all objects of respective dataset. In case objects and NeMOs are rotated analogously, we observe small errors *i.e.* the point clouds are very similar. M-shaped error trajectory, with minimum around 180 degrees of rotation around objects' z-axis, in case of fixed 2D-3D dense representation but rotated ground truth object point cloud. This shows that our representation is aware of rotations.