Zero-shot Inexact CAD Model Alignment from a Single Image
Supplementary Material

Appendix

In this Appendix, we provide additional clarifications, experiments, and results as follows:

6. Additional Implementation Details of Our method

6.1. Training details and hyperparameters

For training the feature adapter, we use the AdamW optimizer [31] with a constant learning rate of $3e^{-4}$, a batch size of 140, and $\tau^{+}_{\text{dist}} = 0.02$, $\tau^{-}_{\text{dist}} = 0.4$, $\tau^{-}_{\text{feat}} = 0.75$, $\alpha = 0.5$, $\beta = 0.1$, $\omega = 0.5$. The dense alignment is optimized with the Adam optimizer at a constant learning rate of 0.005 and $\lambda_{\text{NOC-A}} = 0.33$, $\lambda_{\text{m}} = 3.0$, $\lambda_{\text{d}} = 0.27$.

6.2. Template Rendering

To render templates that cover the entire CAD model, we render each CAD model into 36 templates using Blender, varying 3 elevation angles and 12 azimuth angles. Each angle is randomly sampled from a mean of $[10, 20, 30]^\circ$ for elevation and $[0, 30, 60, \ldots, 330]^\circ$ for azimuth, with a standard deviation of 2, allowing us to capture a broader range of perspectives around the CAD model. The examples of templates are shown in Figure 7.

6.3. Feature Voxel Grid

We used 36 rendered templates with corresponding NOC pairs, augmenting each template into 7 variations (see Appendix 7), resulting in a total of 288 images (36 original + 252 augmented) for constructing a feature voxel grid of size $100 \times 100 \times 100$. Augmented image features were averaged with a weight of 0.071 per image, while rendered images were assigned a weight of 0.5. To smooth the voxel grid, we applied a two-step downsampling and upsampling process using linear interpolation. The grid was first downsampled by factors of 2 and 4, then upsampled back to its original size. The final voxel grid was obtained by a weighted average of these versions, with weights of 0.6 for the original size, 0.25 for the 2× downsampled version, and 0.15 for the 4× downsampled version.

6.4. Feature Adapter Training Data

To generate our training data, we use ShapeNet [7] dataset, which contains a collection of normalized, canonically aligned 3D models. We select models from nine categories that also appear in ScanNet25k [11] (bathtub, bed, bin, bookshelf, cabinet, chair, display, sofa, and table), totaling 2k 3D models. Each model is rendered into 36 templates, following Section 6.2, resulting in 72k rendered training images and NOC pairs. Each template image is further augmented into seven variations and filtered (see Section 7), leading to a final dataset of 300k training images for the feature adapter. Note that we use this feature adapter, which is trained on only nine object categories, to evaluate 20 unseen categories in SUN2CAD.

To validate the design choices, we reserved 2k annotations from the ScanNet training set and used them for validation purposes only.

6.5. Object Mask Generation

For processing ScanNet25k [11] scene images, we use object bounding boxes and retrieved CAD models from ROCA [18]. We then apply Segment Anything (SAM) with ViT-G [13] to generate segmentation masks from the bounding box prompts.

6.6. Metric Depth Estimator

We fine-tuned metric depth estimators on the training images of ScanNet25k [11] and SUN-RGBD [45] in the SUN2CAD dataset for use in comparisons in Section 4. We follow the codebase of DepthAnything [50] by fine-tuning their pre-trained ViT-L relative depth estimator with image-metric depth map pairs from each dataset. The training was performed with a learning rate of $5e^{-5}$ and a batch size of 24.

For ScanNet25k, we used 20k image-depth pairs from the training set, splitting them into 19k for training and 1k for validation. Depth maps were masked to retain only pixels with values > 0.01, and the mask was dilated by 11 pixels to reduce aliasing and noise.

For SUN2CAD, we used 7k images from SUN-RGBD that are in the scene and do not overlap with scenes that appeared in our 550 testing images. We split them into 6.5k for training and 500 for validation. Since SUN-RGBD contains images from multiple sources and cameras, we first normalized all depth maps by resizing them to 518×392 and converting them into canonical inverse depth maps using $C = f/D$ (following [3]), where f is the focal length, D is the metric depth, and C is the inverse depth. We applied the same depth masking and dilation procedure as in ScanNet25k. We then fine-tuned DepthAnything on the masked inverse depth maps. During inference, metric depth maps were reconstructed using the focal length of each test image via $D = f/C$.

In Section 4.2, we use a metric depth estimator that has never been trained on ScanNet [11] dataset for a fair comparison with DiffCAD [16], which is the official DepthAnything [50] Indoor Metric Depth Estimator (ViT-L) trained on NYU dataset [10].

6.7. Visualization of Learned Space

Here we provide implementation details for the feature visualization in Fig. 3 in the main paper. Given a rendered image (Row 1 or 3), we extract a patchwise feature map using either DINOv2 or our feature adapter and obtain its corresponding ground-truth NOC map rom the CAD model we rendered. We then reduce the feature dimensions to two using PCA and visualize them as 2D points in the last two columns. Each feature point is colored based on its corresponding NOC value by mapping (x, y, z) into (r, g, b). The same procedure is applied to real image inputs (Row 2 or 4) using the ground-truth, pose-aligned CAD model to generate the NOC maps.

7. Implementation Details of Rendered Image Augmentation

Renderings often have limited texture and solid color backgrounds, creating a domain gap with real images. We follow DST3D [32] to generate realistic renderings with natural backgrounds, using Stable Diffusion (SD) [42]. While DST3D uses ControlNet [54] to guide image generation with a Canny edge map, we further enhance image fidelity by incorporating multiple visual prompts (Canny edge, depth, and sketch of the renderings) through UniControlNet [55].

However, rarely seen images, such as the back of a cabinet, are challenging for the diffusion model to generate. Such generated images result in artifacts that reduce pose prediction performance. To address this, we propose filtering out poorly generated images based on our NOCS prediction error (See an ablation study in Appendix 12.2). Examples of augmented templates are shown in Figure 8, with filtered-out templates highlighted in red boxes.

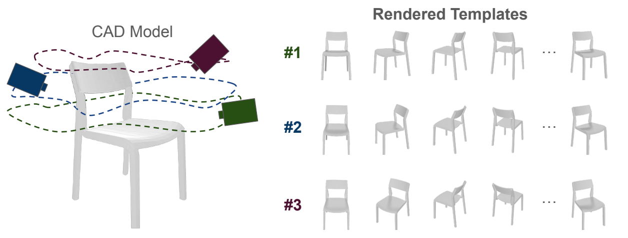

Figure 7. **Rendered templates used in a feature voxel grid and the geometry-aware feature adapter.**

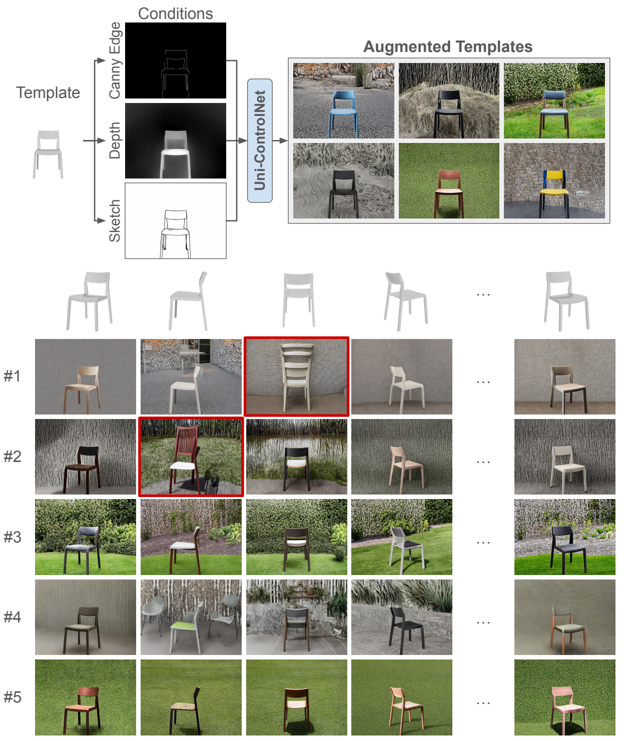

Figure 8. **Augmented templates used in a feature voxel grid and the geometry-aware feature adapter.** Annotated red boxes refer to invalid generated images with a wrong viewpoint shifted from their original renderings on the top row.

8. Implementation Details of Competitors and Ablation Choices

In this section, we provide implementation details for evaluating DiffCAD, FoundationPose, and SPARC in our setting, where scores are computed using their official code. For ROCA, scores are reported as provided in their paper. We also provide details for the dense alignment using features (FM), as presented in the ablation study in Section 4.4.

8.1. DiffCAD

We use DiffCAD's [16] official model weights and code to generate eight pose hypotheses per sample across six categories in their ScanNet25k test subset. We then report the best hypothesis's alignment accuracy in Table 1, denoted as DiffCAD (GT), as a reference to validate our reproduction of their results.

To compute DiffCAD (mean), we calculate translation, scale, and rotation errors for each hypothesis, average them per parameter, and determine alignment accuracy based on standard thresholds used in previous works [18] as: translation error $\leq$ 20 cm, rotation error $\leq 20°$, and scale error $\leq 20\%$ relative to the ground truth.

For DiffCAD (err), we use their codebase to select the best hypothesis based on transformation error on their predicted 2D-3D correspondences. Given X as the set of selected 2D coordinates with depth values, Y_i as the predicted 3D coordinates for the ith hypothesis, and T_i as the solved transformation, the optimal hypothesis is the one minimizing $||Y_i \cdot T_i^T - X||_2$.

8.2. FoundationPose

We use FoundationPose [49]'s officially released model to generate 6-DoF poses on ScanNet25k [11] and SUN2CAD datasets.

For ScanNet25k, we obtain a pair of a 2D bounding box and its corresponding CAD model from ROCA's results. These bounding boxes are processed into 2D object masks using SAM [22], consistent with our method. We also use the same fine-tuned DepthAnything [50] to predict a depth map for each image. To enable fair comparison with our 9-DoF method, we scale the CAD models in two ways: 1) by ground truth scale for the 6-DoF setting and 2) by ROCA's predicted scale for the 9-DoF setting.

For the SUN2CAD dataset, we use a pair of 2D masks and CAD models as input, with depth maps generated by fine-tuned DepthAnything on the SUN RGB-D training split. As with ScanNet25k, we scale the CAD models by their ground-truth scale for the 6-DoF setting.

8.3. SPARC

We follow SPARC [26]'s officially released model weights and code to produce 9-DoF alignment prediction of the SUN2CAD dataset. We utilize depth maps from fine-tuned DepthAnything [50] on SUN RGB-D [45], along with 2D bounding boxes and CAD model pairs from the dataset, as used by other competitors.

In addition to RGB-D images and detected objects, SPARC requires surface normals for both the RGB images and CAD model point clouds. We sample 1,000 surface points and their corresponding normals from each CAD mesh using Trimesh. For image normals, we use Metric3Dv2 [20], a state-of-the-art zero-shot normal estimator, to generate them for all images.

We follow the inference procedure instructed in the paper by initializing the translation using the x and y position at the middle point of a 2D bounding box and $z = 3$. We apply median scaling of the 3D model's category and vary azimuth rotation across $[0, 90, 180, 270]°$ with a fixed elevation angle of $15°$, resulting in 4 hypotheses. Each hypothesis is refined 3 times, and the best hypothesis is selected using their trained classifier for further pose evaluation.

8.4. Feature-based Dense Alignment

This section provides detailed information on dense alignment using DINOv2 features for ablation studies in Section 4.4. The method is inspired by the *featuremetric optimization* refinement method from FoundPose [37]. The goal of featuremetric optimization (FM) is to refine initial 2D-3D correspondences by minimizing projection errors using foundation features. This addresses the misalignment of large 14×14 patch-wise features, whose derived 2D coordinates (e.g., the patch center) may not precisely correspond to their 3D counterparts.

To mitigate this, the method optimizes pose by minimizing discrepancies between features at 3D coordinates (from a CAD model's rendered templates) and their corresponding 2D projections in the input image's feature map. The objective function is:

$$\sum_{(x_i, p_i) \in T_t} \rho\left(p_i - F_q\left(\pi(x_i)\right)\right), \tag{6}$$

where ρ is a cost function, $(p_i, x_i) \in T_t$ is a 3D coordinates and its paired feature, F_q is a 2D feature map, and π is a learnable 2D projection function. Following the original algorithm, we reimplement the method using our feature voxel grid to represent 3D model features. We initialize π with the coarse alignment pose and tune other hyperparameters for optimal performance. We employ PyTorch3D [40] and optimize the loss using Adam with a learning rate of 0.005. We use L1 loss as the cost function and bilinear interpolation for 2D feature sampling via grid sampling.

9. Additional Details on SUN2CAD Dataset

To evaluate our alignment method on unseen or less common object categories, we establish a new inexact match 9-DoF pose alignment test set spanning 20 categories with 550 images.

The primary challenge in aligning images with 3D models is the ambiguity in z-axis translation and scaling, which persists even with manual alignment. We address this using 3D bounding box annotations from the SUN-RGBD dataset [45], which provides real RGB-D images of room scenes derived from 3D scans. These bounding boxes were used as our initial pose, as the bounding box centroid will be a translation, the bounding box size will be used as a scale, and the annotated bounding box orientation will be treated as a rotation.

To map between the 3D model and 2D image, we select LVIS categories [19] that are contained in both SUN-RGBD and CAD models from Objaverse [12] or ShapeNet [7]. For each SUN-RGBD object, we manually choose the most suitable CAD model of the same category and render it to fit the 3D bounding box, then refine its pose by manually rotating the 3D model for more precise alignment. To minimize annotation errors, we exclude objects with incorrect 3D bounding boxes or insufficient edge cues for alignment. We also label object symmetry types following the ScanNet25K baseline: asymmetry, 2-side symmetrical, 4-side symmetrical, and all-side symmetrical.

LVIS category selection is prioritized based on 2D object-CAD similarity, frequency in SUN-RGBD scenes, and object categories with varied and distinct shapes or parts that are different from ScanNet25k's 9 original categories. Initially considering over 50 categories, we finalize 20 categories: basket, bicycle, blender, broom, clock, coffee_maker (coffmkr), crib, fire_extinguisher (fireext), keyboard (keybrd), ladder, lamp, mug, piano, printer, remote_control, shoe, telephone, toaster_oven, vase, and water_bottle. The quantity of each category is shown in Table 2.

Additionally, we aim to simulate a real-world scenario where a user selects a CAD model and object category, and the system automatically aligns it with the scene. Instead of relying on high-quality 2D segmentation masks provided by SUN-RGBD, we predict the segmentation using Grounded-SAM [41], which enables automatic extraction of segmentation masks based on object category.

The samples of SUN2CAD dataset are shown in Figure 20, Figure 21, and Figure 22.

10. Additional Experimentsal Results

10.1. Additional Results on Experiment 4.1

We present a detailed comparison of translation, rotation, and scaling accuracies across competitors on the ScanNet25k dataset in Table 4. All accuracies are computed using the same thresholds defined in Section 4.

In the weakly supervised 9-DoF setting, our method outperforms the 9D adapted FoundationPose [49], designed for exact 6-DoF pose matching, in both translation and rotation for 7/9 categories, achieving better average parameter accuracies. When compared to supervised 9-DoF, ROCA [18], we achieve better translation accuracy in 8/9 categories and rotation accuracy in 4/9 categories. However, when compared to SPARC [26], the strongest supervised baseline, our method exceeds performance in only 2/9 categories for both translation and rotation, revealing a remaining performance gap. Notably, the two supervised methods benefit from either learning object scaling during training or incorporating median object scaling during initialization, whereas our method does not rely on such priors.

In the weakly supervised 6-DoF setting, our adapted 6D solution surpasses FoundationPose [49] in translation for 7/9 categories and in rotation for 8/9 categories, also achieving higher mean accuracy for both parameters.

For a detailed comparison with DiffCAD [16], the state-of-the-art weakly supervised 9-DoF estimator, we present results in Table 5. Our method outperforms DiffCAD (mean) in translation and rotation for 5/6 categories and scaling for 4/6 categories. Additionally, we exceed DiffCAD (Err) in translation, scaling, and rotation for 4/6 categories. Overall, our approach achieves the highest average accuracy across all parameters.

10.2. Additional Results on Experiment 4.3

We also present a detailed comparison of each transformation parameter across competitors on the SUN2CAD dataset in Table 6. Alignment accuracies are computed using the same thresholds defined in Section 4. We also provided the mean and standard error statistics of each parameter for comparison. In the 9-DoF setting, we beat DINOv2 in both accuracies and errors for all parameters. We also beat SPARC [26] in translation and scaling accuracy, but for rotation, we have a lower average rotation error by -7.9% with p-value $= 0.0219$, which is significant. These results indicate that SPARC struggles with the translation and scaling of unseen objects, while it remains somewhat effective in rotation but not as much as our method. While in the inexact 6-DoF setting, we outperform FoundationPose [49] in all metrics.

10.3. Comparison on Unseen Categories in ScanNet25k

We removed one of the nine categories in ScanNet25k [11] used for training and tested single-view accuracies [16] on this removed category to assess the robustness of unseen categories. To do a comparison, we introduce a new baseline called *Real-data NOC regressor (NOC-R)*. NOC-R is similar to NOC-S, except it is trained on real indoor scenes from ScanNet25k [2] dataset, with ground-truth NOC maps generated from CAD models and their pose annotations. This baseline represents what happens if a category-specialized model faces unseen categories.

Table 7 shows that our method experiences only a small drop of $\{-0.4\%, -0.8\%\}$ compared to our variant that trained on all categories, and it still outperforms DINOv2 [36] by $\{+5.6, +5.0\%\}$. In contrast, the strong supervised baseline NOC-R suffers a significant drop by $\{-19.2\%, -24.2\%\}$. Figure 9 visualizes the differences in NOC map predictions for seen vs. unseen categories, highlighting wrong predicted distributions in unseen NOC-R compared to our unseen model.

Pose	Sup	Method	bathtub			bed			bin			bkshlf			cabinet		
			Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro
9D	✓	ROCA [18]	48.9	**77.8**	55.6	21.2	**87.9**	51.5	56.4	**76.0**	**57.3**	21.3	46.2	66.5	19.3	59.8	64.2
	✓	SPARC [26]	**51.6**	69.6	**68.5**	**40.0**	80.0	56.9	62.8	71.3	55.3	**22.7**	**56.6**	**68.7**	**27.1**	**64.5**	**68.9**
	⋆	FoundationPose [49] (for 9D)	35.1	**77.8**	37.4	29.2	**87.9**	36.9	61.5	**76.0**	38.5	6.5	46.2	5.5	6.8	59.8	7.2
	⋆	**Ours**	43.8	51.7	39.3	29.2	46.2	**61.5**	**67.3**	70.8	34.6	22.3	38.3	52.3	20.2	39.7	55.1
6D	⋆	FoundationPose [49]	**51.1**	-	37.8	27.7	-	30.8	64.7	-	30.8	13.9	-	12.4	8.0	-	10.4
	⋆	**Ours** (for 6D)	48.9	-	**38.6**	**29.7**	-	**59.4**	**69.8**	-	**40.1**	**23.6**	-	**46.7**	**26.3**	-	**49.0**

Pose	Sup	Method	chair			display			sofa			table			Avg.		
			Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro
9D	✓	ROCA [18]	45.2	**93.6**	57.9	43.8	**62.2**	51.6	27.8	59.8	63.9	21.5	69.0	**61.8**	36.0	76.8	59.5
	✓	SPARC [26]	**57.7**	93.2	**71.5**	**48.6**	54.7	45.8	40.2	75.2	**79.4**	**27.5**	**74.6**	59.5	**44.8**	**78.3**	**65.6**
	⋆	FoundationPose [49] (for 9D)	46.5	**93.6**	48.3	34.4	**62.2**	36.1	22.7	59.8	36.1	26.9	69.0	42.3	35.1	76.8	37.4
	⋆	**Ours**	57.3	76.7	61.9	44.3	40.9	**53.4**	**47.4**	**77.9**	73.7	24.3	47.4	44.8	43.2	60.3	54.2
6D	⋆	FoundationPose [49]	48.4	-	50.6	52.5	-	40.2	37.1	-	44.3	**32.8**	-	**40.9**	39.7	-	40.2
	⋆	**Ours** (for 6D)	**61.3**	-	**65.6**	**67.2**	-	**68.4**	**54.6**	-	**77.4**	27.7	-	38.3	**48.2**	-	**54.8**

Table 4. **Detailed comparison on ScanNet25k [11] across 9 object categories.** Tr, Sc, and Ro represent translation, scaling, and rotation alignment accuracy. Supervised and weakly supervised baselines are marked with '✓' and '⋆' respectively.

Method	bed			bkshlf			cabinet			chair			sofa			table			Avg.		
	Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro	Tr	Sc	Ro
DiffCAD (Mean)	5.9	35.9	43.8	6.0	10.8	69.6	11.8	35.4	54.9	38.5	51.0	68.5	11.5	34.5	78.9	8.7	12.4	39.5	20.2	33.4	59.0
DiffCAD (Err)	**11.1**	**39.8**	56.9	**22.5**	25.8	**80.1**	19.2	**36.7**	78.5	47.9	54.4	74.5	21.8	46.7	82.3	13.9	23.9	**56.9**	28.6	40.5	70.6
Ours	6.7	31.9	**58.5**	13.5	**35.3**	67.3	**22.7**	31.5	**85.2**	**54.5**	**79.6**	**80.3**	**27.0**	**72.5**	**91.5**	**16.4**	**37.9**	49.6	**32.3**	**56.6**	**71.5**

Table 5. **Detailed comparison on DiffCAD's ScanNet25k split across 6 object categories.** Tr, Sc, and Ro represent translation, scaling, and rotation alignment accuracy.

10.4. Comparison on the Zero-Shot Capability of Feature Adapter

We evaluate the zero-shot capability of geometry-aware features on unseen categories in the SUN2CAD dataset. In addition to Table 2 in the main paper, we report alignment results using only pure geometry-aware features without fusion (Ours ($\omega = 1$)), to assess whether our adapted features can inherit the generalizability of DINOv2 without being explicitly fused. As shown in Table 8, Ours ($\omega = 1$) outperforms DINOv2 by $\{+2.9\%, +7.0\%\}$, demonstrating that our standalone features generalize well to unseen categories and surpass DINOv2 in the inexact CAD model alignment task. Nevertheless, our proposed fused feature version (Ours) still achieves the best performance overall.

10.5. Comparison on Generalizability on Non-textured CAD Models

We evaluate the texture sensitivity of our method by removing textures from textured CAD models and testing single-view alignment accuracies [16] against models with textures. As shown in Table 9, our method shows a minimal drop of $\{-0.17\%, -0.05\%\}$ in mean accuracies, while DINOv2 got a significant decrease of $\{-0.98\%, -1.68\%\}$ mean accuracies. This represents our method's robustness to the textureless CAD model over vanilla foundation features.

10.6. Comparison on Robustness to Model Inexactness

In Figure 10, we assess our method's sensitivity to input CAD model inexactness by replacing the ground-truth model with various models from the same category and reporting accuracy as a function of inexactness, measured by Chamfer distance. (We use rotation accuracy since their shared frontal alignment ensures the same optimal rotation, while shape differences affect translation and scale.) As inexactness increases, our method degrades far less than FoundationPose [49] (-0.14 vs -0.49), the only self-supervised SOTA testable here; DiffCAD [16] predicts pose before CAD retrieval and cannot take an arbitrary model.

10.7. Additional Comparison with FoundationPose

To ensure a fair comparison with FoundationPose [49] under its original assumption used in its 6-DoF setting, we evaluate it using ground-truth depth maps (FoundationPose + GT Depth) and our previously introduced ground-truth object scaling. Table 10 shows a comparison of the ScanNet25k dataset. When given ground-truth depth maps, FoundationPose + GT depth achieves better performance than its default one (FoundationPose) with predicted depth maps for all categories. Our method, which used predicted depth, still surpasses their ground-truth depth version in 5/9 categories and in mean accuracies for both 9-DoF and 6-DoF settings by $\{+1.4\%, +1.7\%\}$. and $\{+2.6\%, +3.0\%\}$,

Pose	Sup	Method	Avg Acc. ↑ Tr	Sc	Ro	Avg Err. ± SE ↓ Tr	Sc	Ro
9D	✓	SPARC [26]	27.82	31.53	**42.00**	0.4381 ± 0.0179	61.26 ± 2.79	53.12 ± 2.37
	-	DINOv2	55.39	32.17	24.31	0.2698 ± 0.0152	68.65 ± 5.70	62.88 ± 2.08
	⋆	**Ours**	**62.36**	**43.09**	41.09	**0.2374 ± 0.0134**	**41.17 ± 2.27**	**46.03** ± 1.98
6D	⋆	FoundationPose [49]	55.81	-	26.72	0.2778 ± 0.0174	-	94.88 ± 2.88
	⋆	**Ours** (for 6D)	**64.72**	-	**41.63**	**0.2235 ± 0.0134**	-	**44.61 ± 1.97**

Table 6. **Comparison in single-view accuracy on SUN2CAD dataset.** Tr, Sc, and Ro represent translation, scaling, and rotation alignment accuracy/errors. Supervised, weakly supervised, and unsupervised baselines are marked with '✓', '⋆', and '-' respectively.

Seen	Method	Sup	bathtub	bed	bin	bkshlf	cabinet	chair	display	sofa	table	Cat.↑	Inst.↑
✓	NOC-R	✓	**38.8**	**22.3**	**33.0**	**18.7**	**40.4**	**46.5**	**32.1**	**46.8**	**24.6**	**33.7**	**37.6**
	Ours	⋆	31.6	13.7	18.5	13.2	20.5	40.0	24.7	39.5	22.8	24.9	30.1
×	NOC-R	✓	18.5	0.0	7.7	10.0	**28.3**	12.9	12.4	31.9	8.4	14.5	13.4
	DINOv2	-	22.7	10.2	14.9	5.8	18.7	34.9	12.6	34.8	15.9	18.9	24.3
	Ours	⋆	**28.6**	**14.2**	**21.7**	**12.9**	23.5	**38.3**	**20.9**	**37.6**	**22.2**	**24.5**	**29.3**

Table 7. **Comparison in single-view accuracy on unseen categories on ScanNet25k.** Supervised, weakly supervised, and unsupervised baselines are marked with '✓', '⋆', and '-' respectively. Our method maintains accuracy on unseen classes while the supervised baseline NOC-R completely fails to adapt.

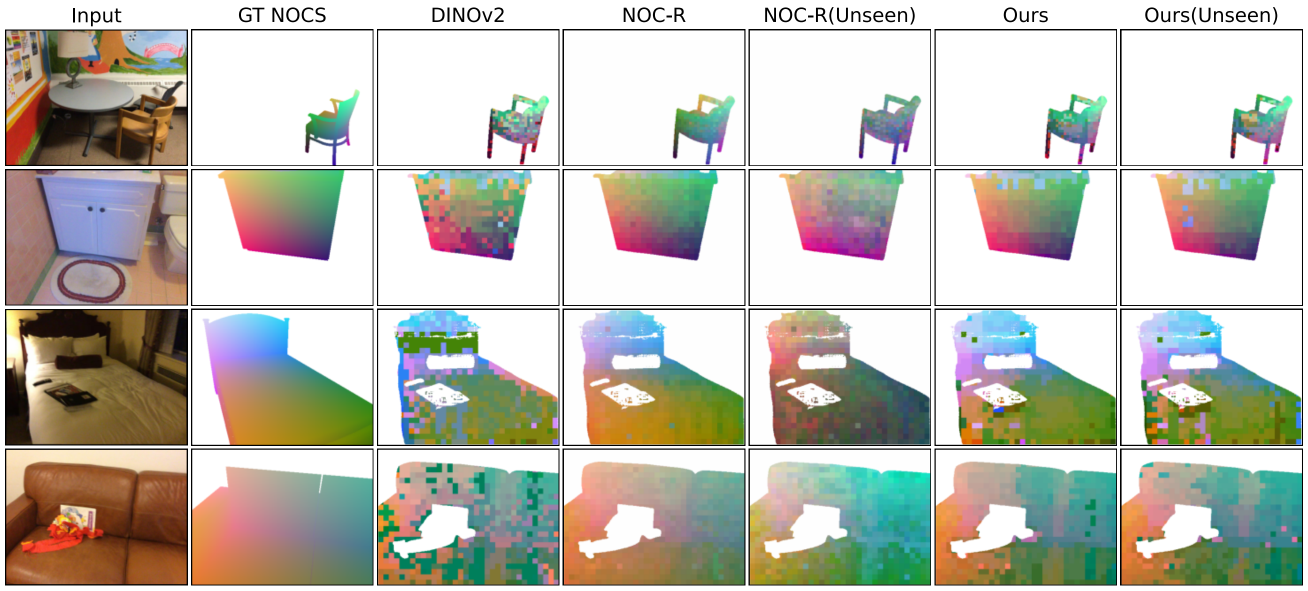

Figure 9. **Qualitative results in NOC map prediction on unseen ScanNet25k categories.** Our method reliably generates more accurate NOC maps than DINOv2 and is robust against unseen categories, unlike NOC-R, which excels in standard settings but struggles in unfamiliar categories.

respectively. For a comparison in SUN2CAD dataset shown in Table 11, our adapted method surpasses FoundationPose with ground-truth depth 9/20 categories and tied in 3 categories. Although we still outperform them in mean accuracies by $\{+5.3\%, +1.2\%\}$.

11. Additional Qualitative Results

11.1. Qualitative Results in NOC Map Prediction

Figure 11 and Figure 12 show the NOC prediction results of our method and other NOC predictor baselines, NOC-S and DINOv2, on ScanNet25k [11] and SUN2CAD, re-

Group	Method	basket #7	bicycle #14	blender #7	broom #2	clock #13	coffmkr #19	crib #18	fireext #15	keybrd #66	ladder #4	lamp #132	mug #59	piano #18	printer #92	remote #8	shoe #3	phone #47	oven #14	vase #9	bottle #3	Cat.↑ #20	Inst.↑ #550
9D	DINOv2	0.0	**28.6**	**14.3**	**50.0**	**7.7**	10.5	0.0	**13.3**	3.0	0.0	5.4	0.0	44.4	7.6	0.0	**33.3**	6.4	7.1	0.0	0.0	11.6	7.3
	Ours (ω=1)	14.3	14.3	0.0	**50.0**	**7.7**	**26.3**	5.6	6.7	12.1	0.0	**9.8**	**11.9**	**50.0**	19.6	0.0	0.0	**8.5**	42.9	11.1	0.0	14.5	14.3
	Ours	**42.9**	21.4	**14.3**	**50.0**	**7.7**	21.1	**16.7**	6.7	**24.2**	**25.0**	6.1	10.2	**50.0**	**25.0**	**25.0**	**33.3**	**8.5**	**57.1**	11.1	**33.3**	**24.5**	**17.6**

Table 8. **Comparison in single-view accuracy on the zero-shot capability of feature choices on SUN2CAD dataset.** Our fused geometry-aware features achieve the highest accuracy. Even without fusion, pure geometry-aware features (Ours (ω=1)) outperform DINOv2, demonstrating inherited generalization to unseen categories.

Method	Texture	bathtub	bed	bin	bkshlf	cabinet	chair	display	sofa	table	Cat.↑	Inst.↑
DINOv2	✓	30.77	7.43	14.65	5.97	18.63	33.71	13.64	35.12	14.77	19.14	24.13
	×	28.71	6.61	14.58	4.26	17.63	30.56	11.30	35.44	14.40	18.16	22.45
Ours	✓	35.89	10.74	19.89	14,48	21.46	38.53	23.02	39.87	21.90	25.09	29.81
	×	34.35	8.26	26.17	13.24	22.62	38.49	23.02	36.07	22.04	24.92	29.76

Table 9. **Comparison in single-view accuracy on removing CAD model texture on ScanNet25k.** Our method retains similar accuracies when removing a query CAD model's textures, compared to DINOv2, which shows more significant accuracy drops.

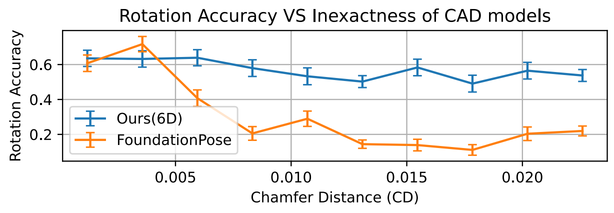

Figure 10. Rotation accuracy under varying CAD model inexactness, measured by Chamfer distance and averaged over 100 ScanNet25k images.

spectively. Our method produces smoother, more accurate NOC maps, whereas DINOv2 exhibits noise due to object symmetry (e.g., left vs. right). While NOC-S produces a shifted NOC distribution in some samples, our method did not suffer from this problem due to direct correspondence matching between a given 3D model and an input image.

11.2. Qualitative Results in Fine Pose Optimization

Figure 13 illustrates a comparison between poses from our coarse alignment prediction and refined ones from our fine alignment pipeline in ScanNet25k images. The results suggest an improvement in both rotation and object placement to match the appearance of the object in the input images.

11.3. Qualitative Results in ScanNet25k Images

We present comparison results on ScanNet25K for entire-scene CAD alignment, evaluating against all our baselines: 9-DoF supervised, 9-DoF weakly supervised, and 6-DoF weakly supervised methods, with randomly selected results shown in Figure 17, Figure 18, and Figure 19. Please note that the 3D model in ground-truth alignments might be different from each method's retrieved 3D models.

11.4. Qualitative Results in SUN2CAD

Figure 20, Figure 21, and Figure 22 present comparisons in CAD alignment results on random test samples from the SUN2CAD dataset between our method, SPARC [26](9-DoF), DINOv2 (9-DoF) and FoundationPose [49] (6-DoF).

12. Additional Ablation Studies

12.1. Study on the Architecture of Feature Adapter

We study the best network architecture for our feature adapter. We choose 3 choices: Autoencoder (AE), ViT [13] layers, and MLP layers. AE represents CNNs capable of decoding feature maps into pixel-level outputs, aligning with our NOC prediction objective. We follow AE architecture from Stable Diffusion [42]'s VAE decoder. ViT layers specialize in capturing spatial relationships through self-attention, making them well-suited for mapping feature maps to 3D structures in NOC prediction. MLP serves as a simple yet effective linear layer for adapting zero-shot DINOv2 [36] features to a new feature space.

Table 14 reports NOC prediction errors (NOC error) on the ScanNet25k validation set, measured as the RMSE between predicted and ground-truth NOC maps, averaged over object pixels within the segmented mask. Following Section 3.1.1, we construct a feature voxel grid to establish 2D-3D correspondences for NOC prediction. Among the tested architectures, MLP achieves the lowest NOC error, making it the best choice.

Group	Method	bathtub	bed	bin	bkshlf	cabinet	chair	display	sofa	table	Avg Cat.↑	Avg Inst.↑
9D	FoundationPose [49] (for 9D)	20.0	22.9	27.6	0.9	3.1	41.8	23.6	15.0	17.5	19.2	25.7
	[49] (for 9D) + GT depth	**23.3**	**25.7**	**34.5**	1.9	3.1	45.4	20.4	20.4	**20.3**	21.7	28.4
	Ours	16.7	18.6	22.8	**12.7**	**9.2**	**49.3**	**24.1**	**38.1**	16.5	**23.1**	**30.1**
6D	FoundationPose [49]	22.5	21.4	37.5	6.1	5.8	44.5	30.4	29.2	27.1	24.9	31.1
	[49] + GT depth	**25.8**	**27.1**	**47.0**	9.0	7.3	46.7	32.5	38.1	**33.1**	29.6	35.0
	Ours (for 6D)	20.8	25.7	27.6	**19.8**	**22.7**	**56.1**	**51.8**	**45.1**	20.1	**32.2**	**38.0**

Table 10. **Comparison in NMS accuracy [33] on ScanNet25k [2] against FoundationPose [49].** We additionally provide GT depth maps to FoundationPose to match their original setting. Our method still surpasses [49] + GT depth in average alignment accuracies.

Group	Method	basket	bicycle	blender	broom	clock	coffmkr	crib	fireext	keybrd	ladder	lamp	mug	piano	printer	remote	shoe	phone	oven	vase	bottle	Cat.↑	Inst.↑
		#7	#14	#7	#2	#13	#19	#18	#15	#66	#4	#132	#59	#18	#92	#8	#3	#47	#14	#9	#3	#20	#550
6D	FoundationPose [49]	28.6	**50.0**	**14.3**	50.0	7.7	5.3	11.1	33.3	33.3	25.0	23.5	15.3	11.1	7.6	**25.0**	0.0	29.8	21.4	0.0	33.3	21.3	20.4
	[49] + GT depth	28.6	**50.0**	**14.3**	**100.0**	7.7	10.5	11.1	**53.3**	31.8	**75.0**	**47.7**	**27.1**	16.7	9.8	0.0	0.0	**31.9**	0.0	**55.6**	33.3	30.2	29.5
	Ours (for 6D)	**42.9**	**50.0**	**14.3**	50.0	**23.1**	**36.8**	**16.7**	20.0	**39.4**	50.0	30.3	20.3	**61.1**	**27.2**	**25.0**	0.0	13.8	**78.6**	44.4	**66.6**	**35.5**	**30.7**

Table 11. **Comparison in Single-view accuracy [16] on SUN2CAD against FoundationPose [49].** We additionally provide GT depth maps to FoundationPose to better match their original setting. Our method still surpasses [49] + GT depth in average alignment accuracies.

12.2. Study on Training Data for Feature Adapter

Table 12 presents an ablation study on the training data used for our geometry-aware adapter. We evaluate NOC predictions by using the same feature voxel grid construction and 2D feature map with nearest-neighbor matching, while varying the adapter model used to generate the features. The results indicate that using both rendered and generated images leads to higher NOC errors than using solely rendered images due to incorrect synthetic samples, as shown in Figure 8.

To address this issue, we propose filtering out invalid generated images using NOC prediction. Specifically, we estimate the NOC map for each generated image by applying our nearest-neighbor matching with a feature voxel grid from rendered templates, as described in Section 3.1. Here, a generated image will serve as the input feature map instead of a real image. The predicted NOC map is then compared to the ground-truth NOC map to calculate NOC error and assess whether the generated image produces features sufficiently similar to the original rendered templates. We discard generated images with an NOC error exceeding 0.20. Training our feature adapter on filtered and rendered images results in improved performance and the lowest NOC error.

Training data	NOC error ± SE ↓
Rendered	0.2313 ± 0.0022
Generated	0.2412 ± 0.0022
Rend+Gen	0.2330 ± 0.0023
Filtered Rend+Gen	**0.2263 ± 0.0023**

Table 12. **Ablation study in training data of our feature adapter.**

Features	# Rendered	# Augmented	# Total	NOC error ± SE ↓
DINOv2	36	-	36	0.2679 ± 0.0018
	-	36	36	0.2822 ± 0.0018
	144	-	144	0.2613 ± 0.0019
	-	144	144	0.2775 ± 0.0018
	36	108	144	0.2474 ± 0.0020
	36	**252**	**288**	**0.2466 ± 0.0020**
Ours	36	-	36	0.2364 ± 0.0022
	-	36	36	0.2396 ± 0.0022
	144	-	144	0.2335 ± 0.0022
	-	144	144	0.2372 ± 0.0022
	36	108	144	0.2278 ± 0.0023
	36	**252**	**288**	**0.2263 ± 0.0023**

Table 13. **Ablation study on feature voxel grid construction.**

12.3. Study on the Architecture of NOC-S

We conduct an ablation study on our introduced baseline, NOC-S, to optimize its performance for a fair comparison with our method on the ScanNet25k validation set. We tested learning NOC reconstruction objective (Eq. 1) on the same choice of architectures, ViT [13] layer, MLP, and Autoencoder (AE) [42]. Unlike the feature adapter's NOC prediction, which relies on nearest-neighbor matching, NOC-S directly outputs 2D-3D correspondences from a final MLP layer. We hypertuned each choice for optimal performance. The results in Table 15 show that ViT achieves the lowest NOC error.

Our final model consists of a single ViT layer with a one-layer MLP, mapping encoded DINOv2 [36] feature maps to NOC. In addition to feature maps, we also concatenate a query CAD model's global feature, which is gener-

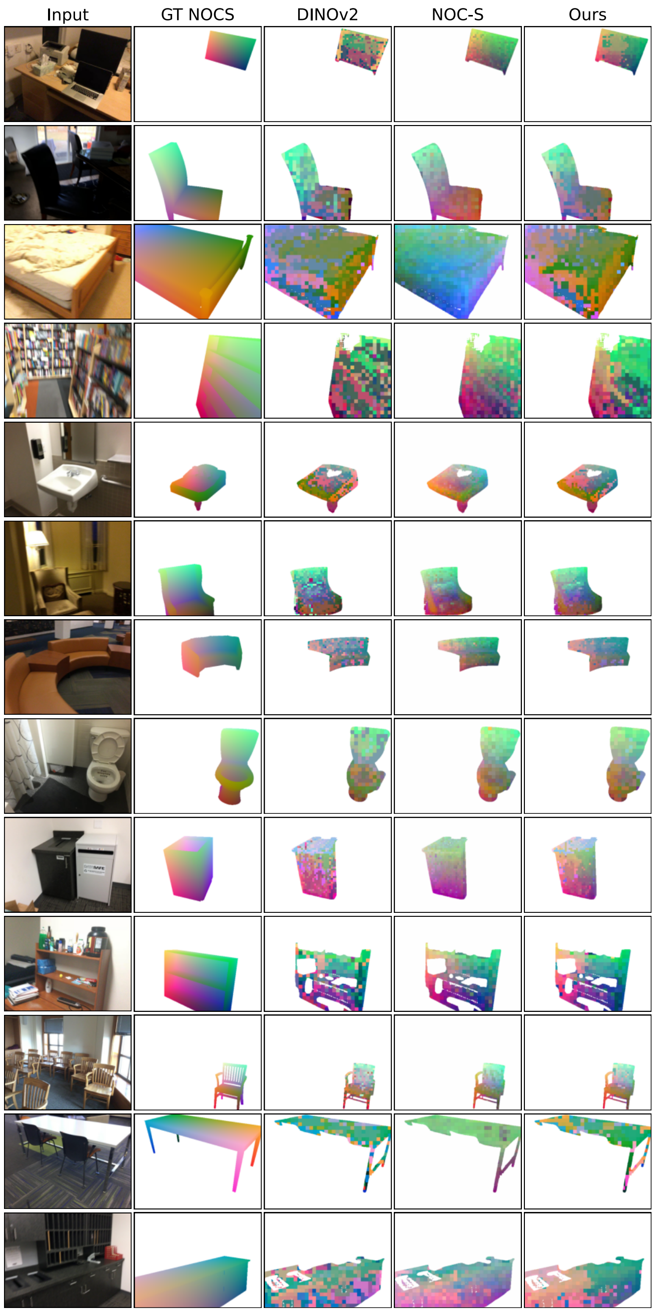

Figure 11. **Qualitative results in NOC prediction on ScanNet25k dataset.** Our solution provides better smoothness and correctness than DINOv2, while NOC-S learns to output a smooth NOC map, multiple samples show shifted or invalid NOC range compared to ground-truth NOC maps.

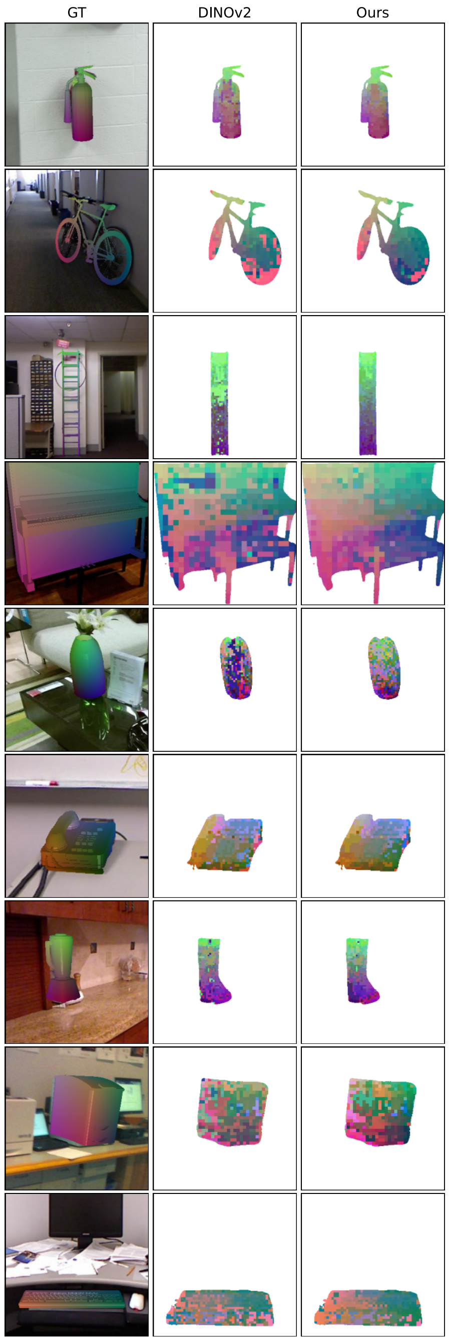

Figure 12. **Qualitative results in NOC prediction on SUN2CAD dataset.** Our solution provides better smoothness and correctness than DINOv2 in the same zero-shot setting.

ated by averaging CLS tokens from all CAD model's rendering templates using DINOv2. This results in a CAD-conditioned feature map with size $\mathbb{R}^{H\times W\times 2048}$. The ViT layer has a hidden size of 2048 with 8 attention heads, while the MLP predicts 3-channel NOC outputs. We train the model using AdamW [31] optimizer using a constant learning rate $5e^{-4}$ and training batch size of 150.

12.4. Study on Feature Voxel Grid and Templates

Table 13 presents the impact of different template types on feature voxel grid construction on ScanNet25k validation

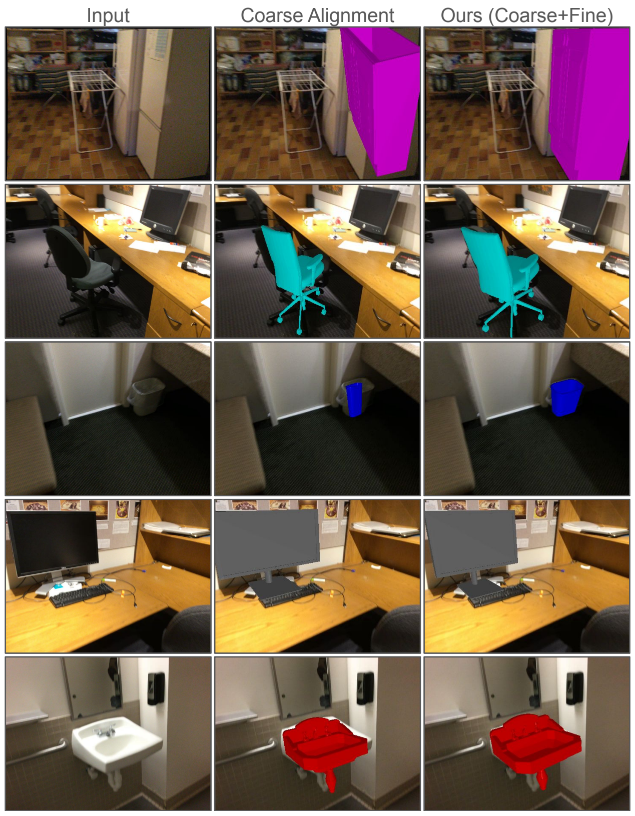

Figure 13. **Qualitative results of dense image-based alignment optimization,** shown before (coarse alignment) and after fine pose optimization.

Architecture	NOC error $\pm$ SE $\downarrow$
AE	0.2485 ± 0.0024
MLP	$\mathbf{0.2263 \pm 0.0023}$
ViT	0.2454 ± 0.0020

Table 14. **Ablation study in architecture choices of our feature adapter.** MLP provides the best NOCS prediction error over other choices.

Architecture	NOC error $\pm$ SE $\downarrow$
AE	0.2094 ± 0.0015
MLP	0.2040 ± 0.0014
ViT	$\mathbf{0.1966 \pm 0.0017}$

Table 15. **Ablation study in architecture choices of NOC-S.** ViT provides the best NOCS prediction error.

set. Using only augmented templates (generated images) results in higher NOC errors than rendered images in the NOC output from feature matching. However, combining both rendered and generated images yields better performance. The improvements are more significant when using DINO features (-0.0309) than our GFS features (-0.0109). Nevertheless, our GFS features result in lower NOC errors than using DINOv2 features alone.

12.5. Study on Triplet Loss Hyperparameters

In addition to the feature adapter loss study of β in Section 4, we present a hyperparameter study on τ^+_{dist}, τ^-_{dist}, and τ^-_{feat} using ScanNet25k validation set on NOC error metric. The results are shown in Figure 14. Higher values of τ^+_{dist} result in larger NOC errors due to selecting features from different object parts as positive pairs. Similarly, excessively high values of τ^-_{dist} and τ^-_{feat} reduce negative sample diversity, increasing NOC error. Likewise, very low τ^+_{dist} values reduce positive pair variation, also leading to higher NOC error.

12.6. Study on Dense Image-based Alignment Hyperparameters

We study the single-view alignment accuracies of our dense image-based alignment hyperparameters $\lambda_{\text{NOC-A}}, \lambda_{\text{m}}$, and λ_{d} on ScanNet25k validation set. To normalize each loss term, we first compute the mean alignment loss using initial poses from our coarse pose estimator. The mean values are $N_{\text{NOC-A}} = 0.33$, $N_{\text{m}} = 1$, and $N_{\text{d}} = 0.067$. Each term is then normalized relative to the smallest mean loss, yielding the scaled hyperparameters: $\lambda_{\text{NOC-A}} = N_{\text{NOC-A}}\lambda'_{\text{NOC-A}}$, $\lambda_{\text{m}} = N_{\text{m}}\lambda'_{\text{m}}$, and $\lambda_{\text{d}} = N_{\text{d}}\lambda'_{\text{d}}$.

For hyperparameter tuning, we fixed $\lambda'_{\text{NOC-A}} = 1$ and perform a grid search for λ'_{m} and λ'_{d}. As shown in Figure 15, the highest alignment accuracy is achieved with $\lambda'_{\text{m}} = 3$ and $\lambda'_{\text{d}} = 4$, resulting in final values of $\lambda_{\text{NOC-A}} = 0.33$, $\lambda_{\text{m}} = 3$, and $\lambda_{\text{d}} = 0.27$.

13. Failure Cases

Figure 16 illustrates failure cases. Poor input image quality, such as small objects (A), degrades DINOv2 features, making it difficult to distinguish object parts correctly, which can lead to incorrect pose estimation.

Failures in dependencies, such as depth prediction or mask segmentation, also affect our method. For example, in (B), the mask prediction merges two bicycles into one. As a result, while our method produces a good NOC map for the target bicycle in the back (C), it also unintentionally includes parts of the front bicycle (D), causing misalignment by aligning with the front bicycle instead.

Objects with minimal edge cues, such as a toaster oven with severe self-occlusion, present another challenge. The NOC map (E) lacks sufficient edge information to infer depth, leading to an incorrect shape prediction.

Alignment ambiguity arises when multiple matched parts have inconsistent spatial relationships between the CAD model and the real object. Our method aims to cover all local correspondences, whereas humans may prioritize

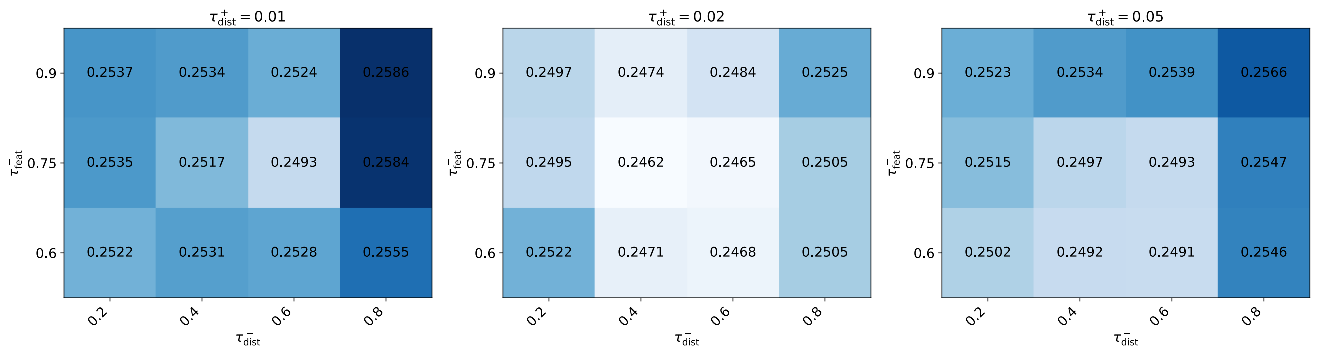

Figure 14. **Ablation study in Triplet loss hyperparameters τ_{dist}^{+}, τ_{dist}^{-}, and τ_{feat}^{-} on NOC error metric.**

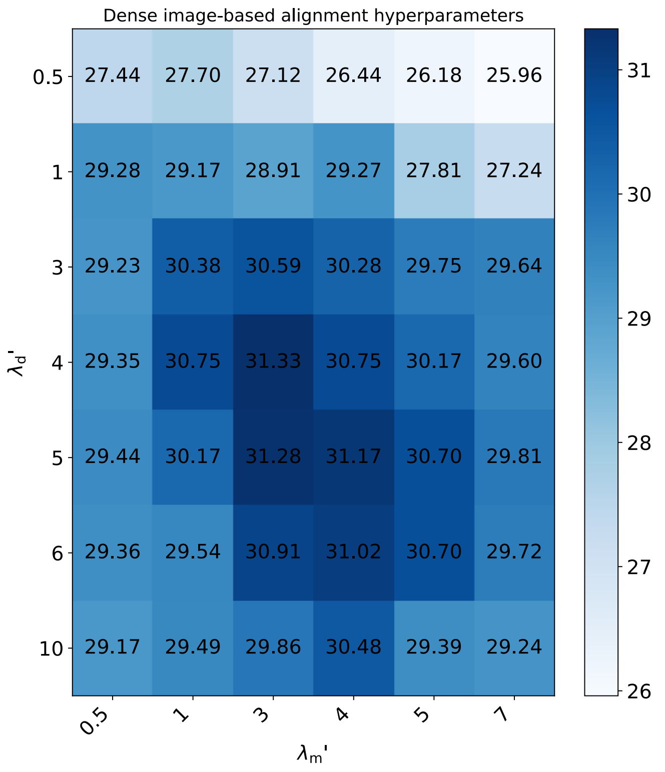

Figure 15. **Ablation study in Dense Image-based Alignment Hyperparameters on Single-view alignment accuracy [16].**

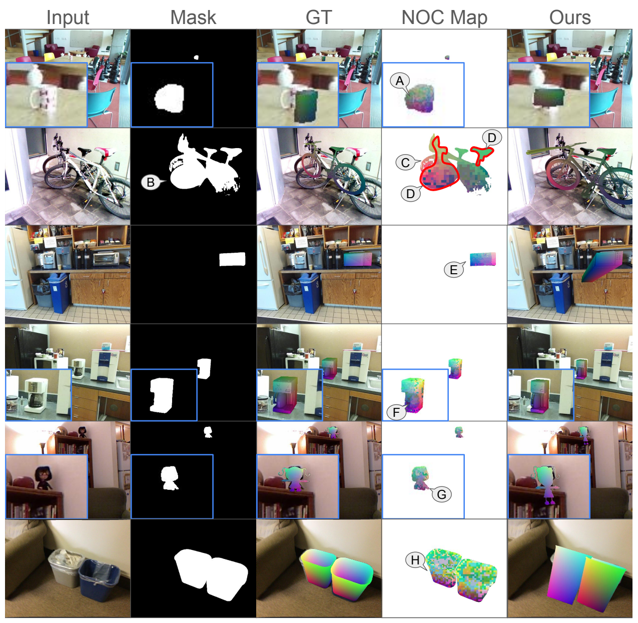

Figure 16. **Failure cases;** (A) Poor input image quality; (B) incorrect input mask, causing two objects' features to blend together (C, D) for CAD alignment; (E) lack of edge cues for inferring object depth; (F, G) alignment ambiguity; (H) Thin object.

specific object parts. In (F), the CAD model of a coffee maker has a kettle on the left, whereas the real object has it in the middle. Our method prioritizes aligning the kettle position rather than focusing on the overall coffee maker shape, deviating from the ground-truth. Similarly, in (G), when aligning a CAD model with significant visual differences from its target object, our method tries to match individual parts, such as the doll's hand, while a human might prioritize aligning the doll's head.

Lastly, although (H) clearly distinguishes between top and bottom, the thin structure complicates distinguishing inside from outside on the same side, making it more prone to failure compared to other objects.

14. Societal Impacts

Our work on 9-DoF pose estimation benefits real-world applications in synthetic environments such as VR and gaming, where safety is not a concern. However, its accuracy may be insufficient for safety-critical tasks like autonomous driving. Reliance on predicted depth alone can introduce translation-scale ambiguity, and further accuracy improvements are needed to mitigate potential risks in high-stakes environments.

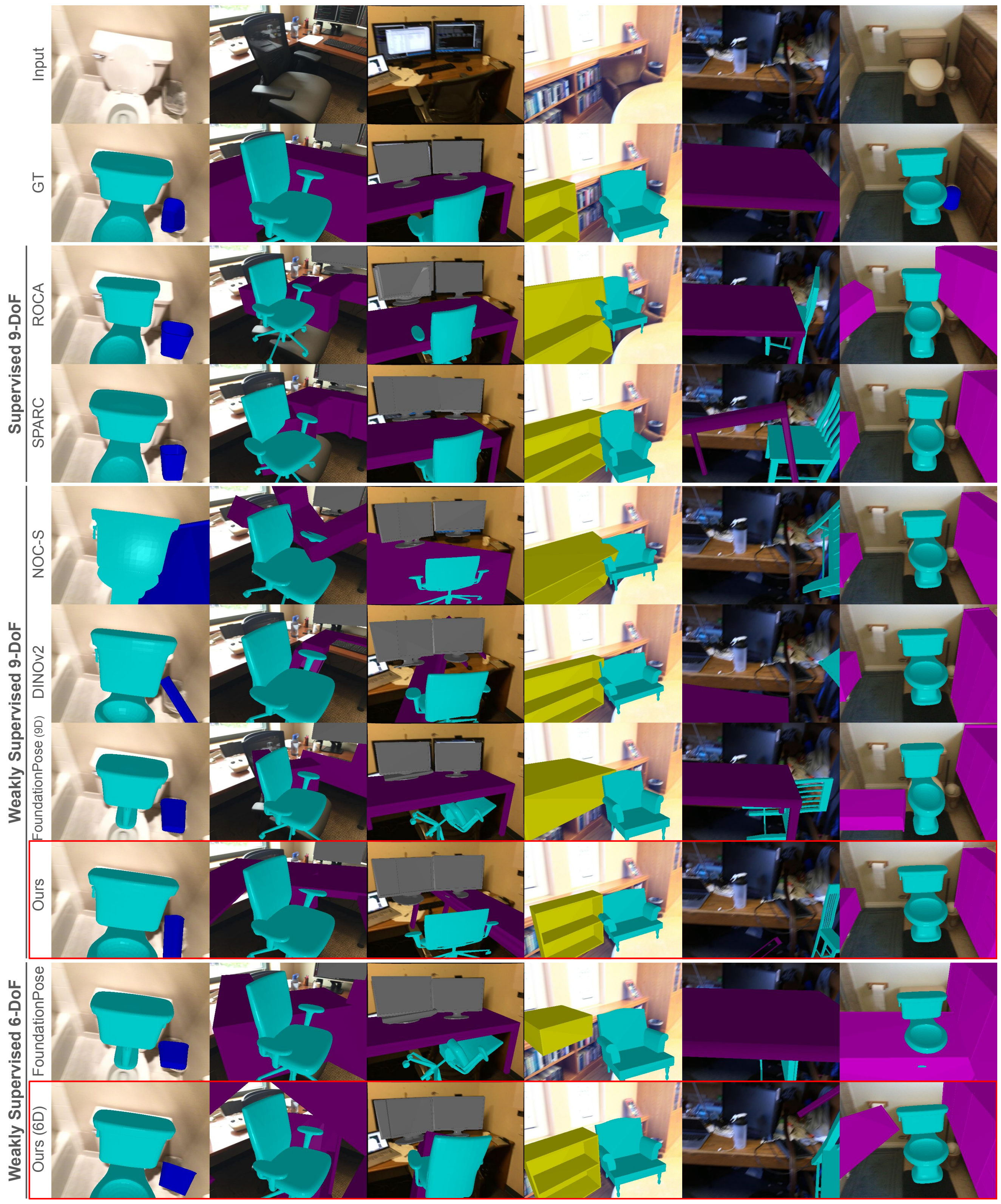

Figure 17. **Random test samples from ScanNet25k.** We compare entire-scene CAD alignment against 9-DoF supervised methods (SPARC [26] and ROCA [18]), 9-DoF weakly supervised methods (DiffCAD [16]), and 6-DoF weakly supervised baselines (Foundation-Pose [49])

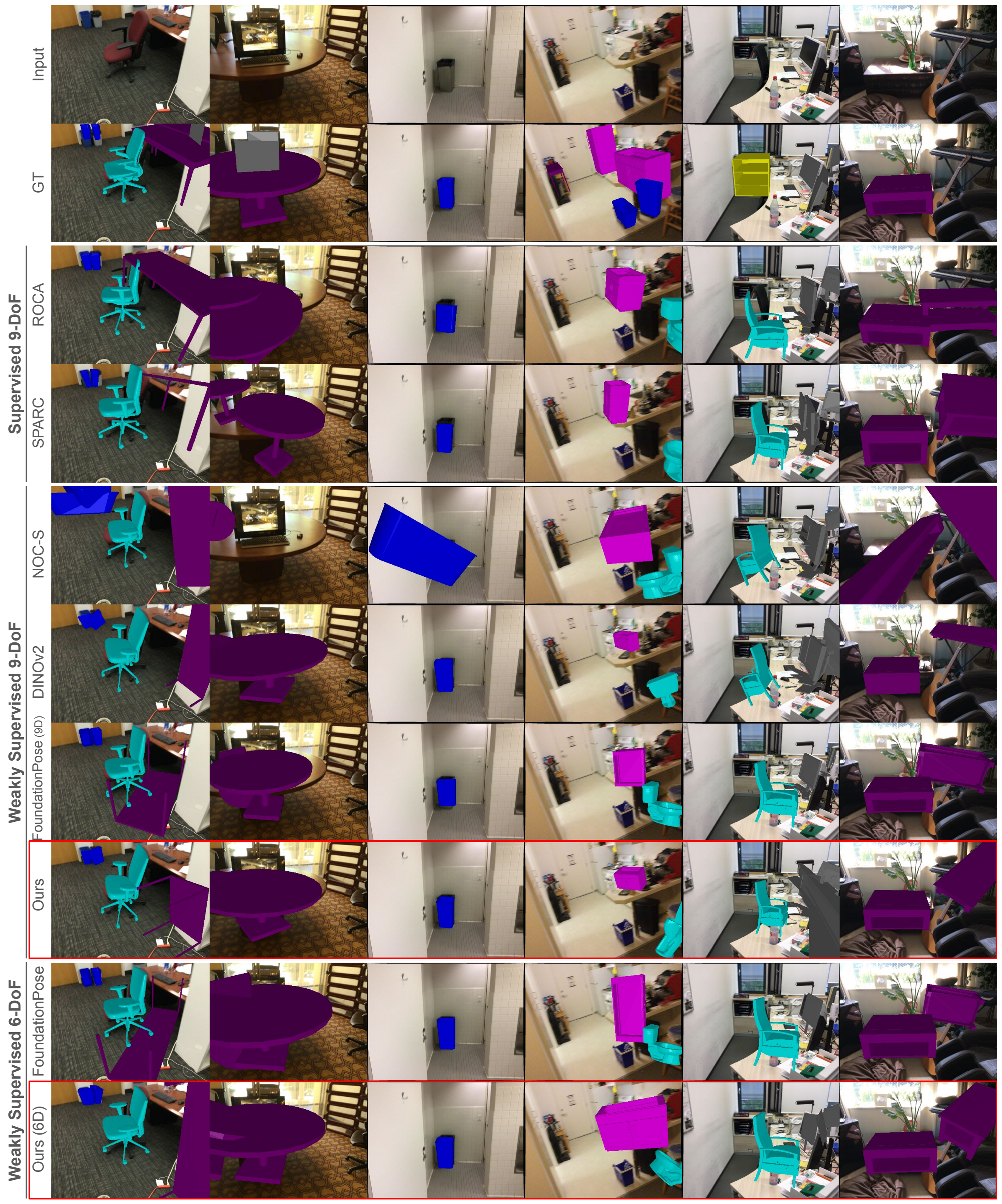

Figure 18. **Random test samples from ScanNet25k.** We compare entire-scene CAD alignment against 9-DoF supervised methods (SPARC [26] and ROCA [18]), 9-DoF weakly supervised methods (DiffCAD [16]), and 6-DoF weakly supervised baselines (FoundationPose [49])

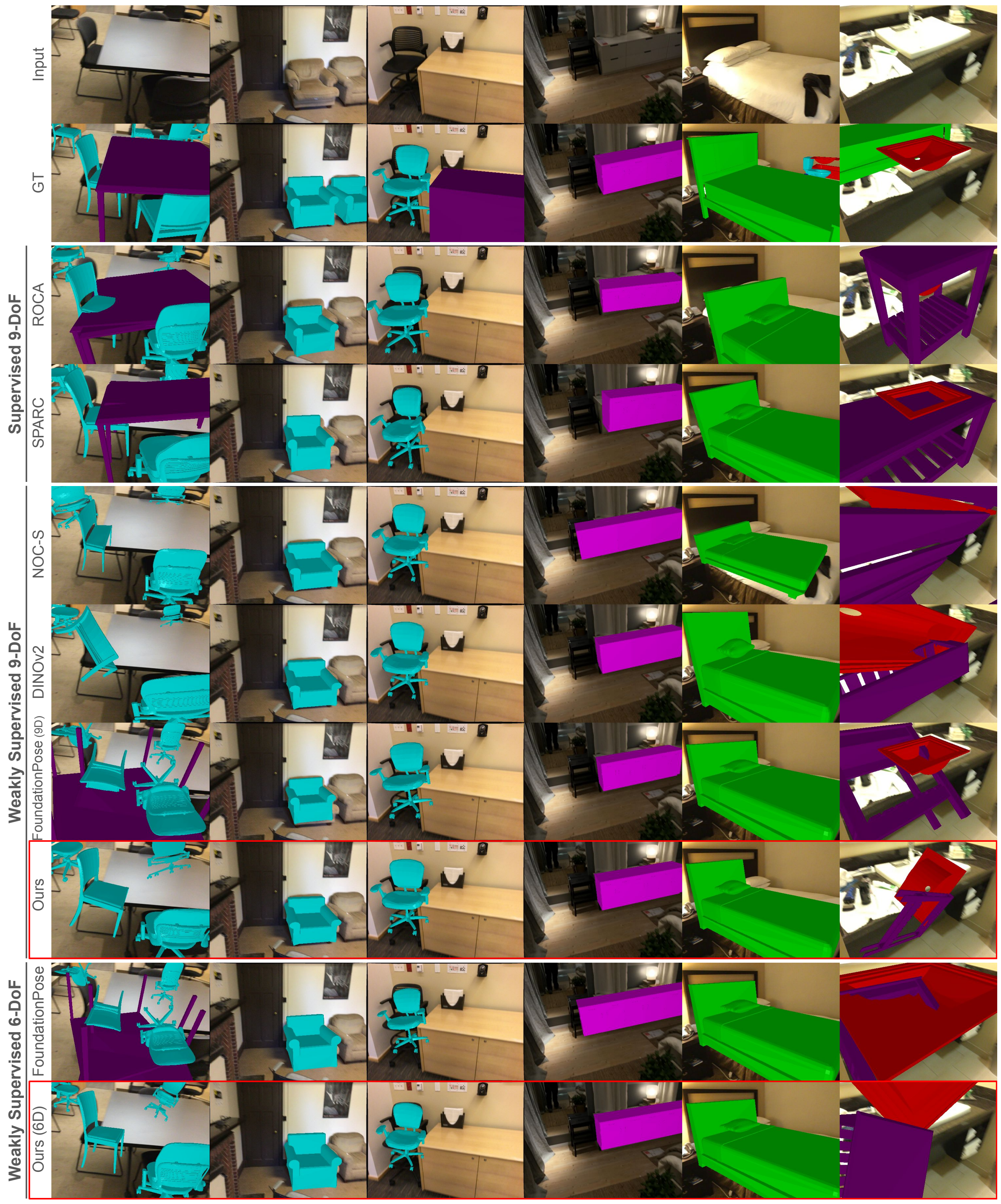

Figure 19. **Random test samples from ScanNet25k.** We compare entire-scene CAD alignment against 9-DoF supervised methods (SPARC [26] and ROCA [18]), 9-DoF weakly supervised methods (DiffCAD [16]), and 6-DoF weakly supervised baselines (FoundationPose [49]).

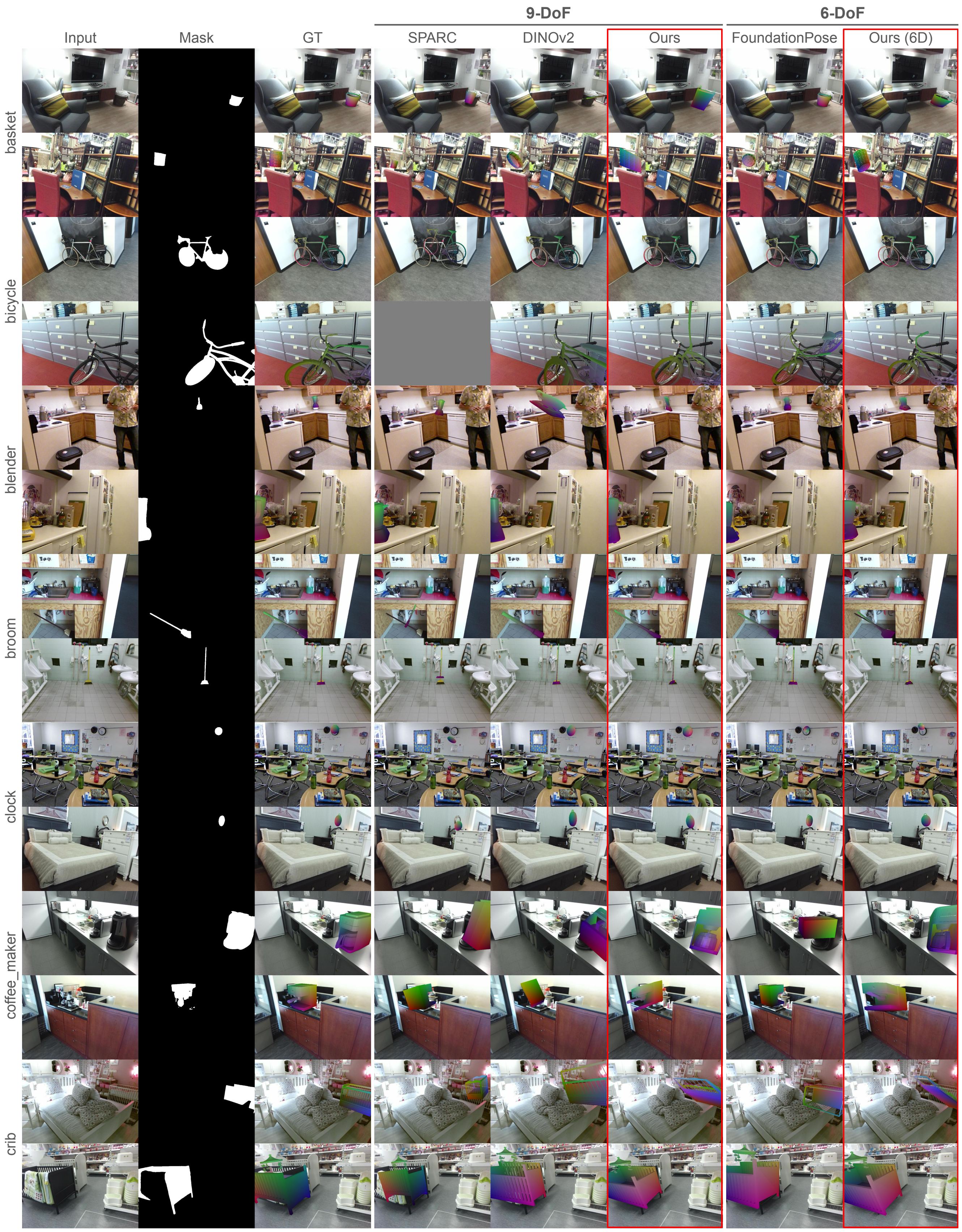

Figure 20. **Random test samples from SUN2CAD** comparing 9-DoF pose predictions with SPARC [26] and DINOv2, and 6-DoF predictions with FoundationPose [49].

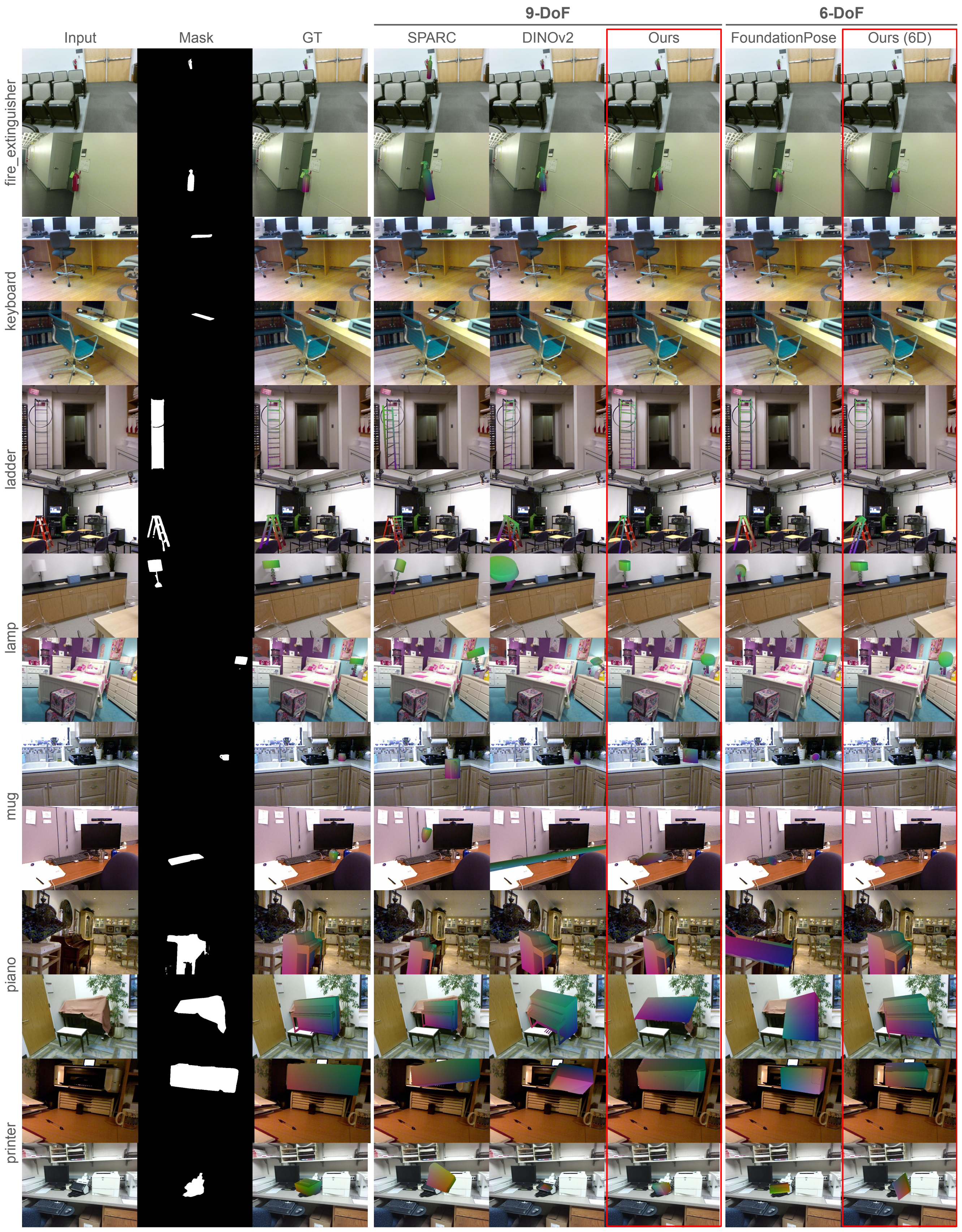

Figure 21. **Random test samples from SUN2CAD** comparing 9-DoF pose predictions with SPARC [26] and DINOv2, and 6-DoF predictions with FoundationPose [49].

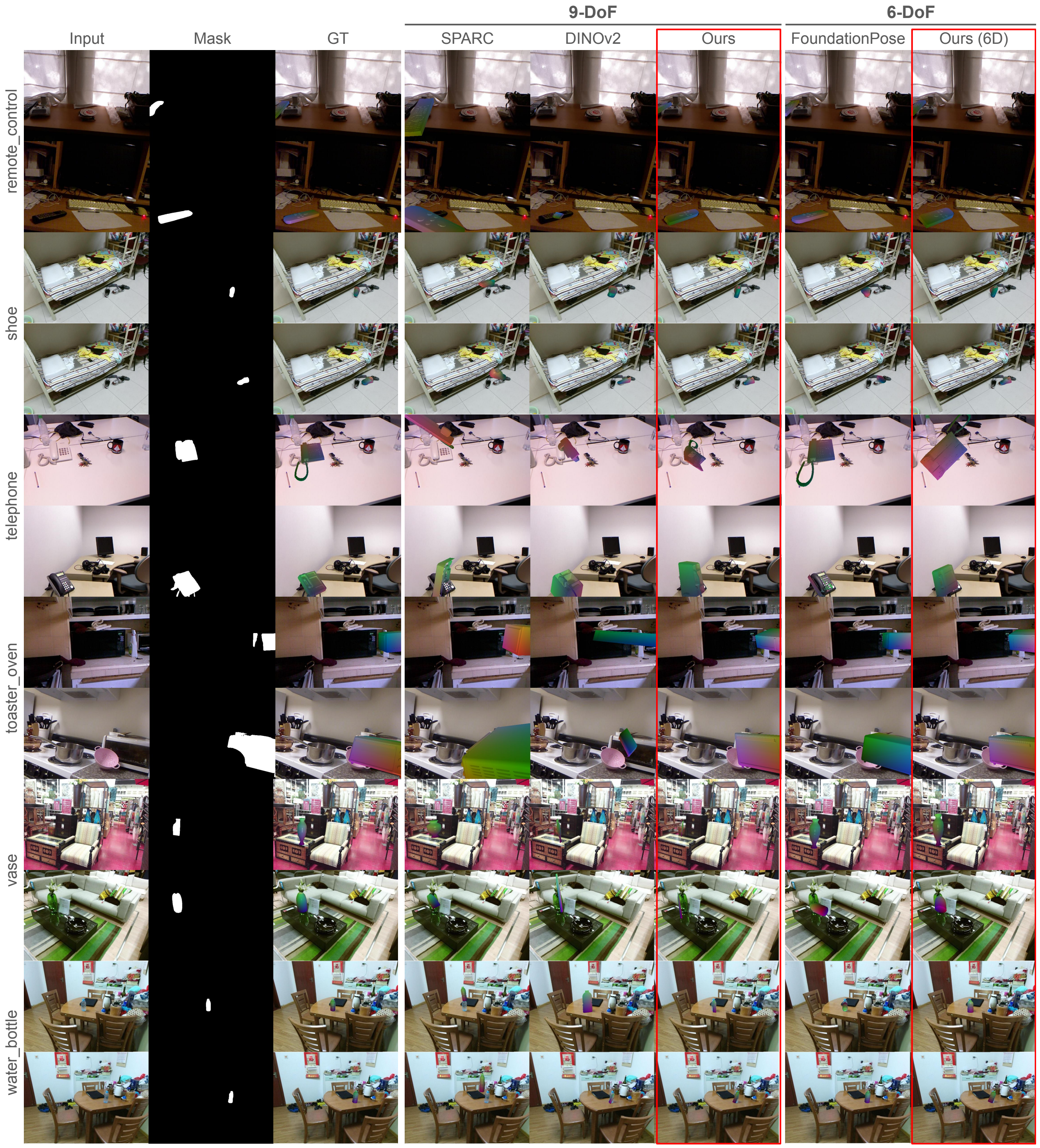

Figure 22. **Random test samples from SUN2CAD** comparing 9-DoF pose predictions with SPARC [26] and DINOv2, and 6-DoF predictions with FoundationPose [49].