Object Pose Transformer: Unifying Unseen Object Pose Estimation

Supplementary Material

We first provide an overview of the taxonomy of the current unseen object pose estimation in Tab. 6.

1. More Method Details

We focus on more method details that are not fully covered in the main paper. We describe how keypoints are instantiated and refined, how NOCS is realized via FiLM-conditioned bin classification, and how the pose- and scale-regression heads are constructed.

1.1. Keypoint Detection and Fusion

Instantiated keypoint module. The keypoint detector in OPT-Pose is a small Transformer-style module on top of sampled point features. For each frame, we start from a fixed set of M learnable query embeddings $Q \in \mathbb{R}^{M\times D}$ that are shared across the batch. Given the concatenated visual–geometric descriptors $F \in \mathbb{R}^{B\times N\times D}$ at N sampled pixels per frame (cf. main paper), the detector performs the following steps:

1. **Cross-attention.** We first apply LayerNorm to the query and use it as the input to a multi-head attention layer as both key and value. This step produces attention weights over K sampled pixels for each of the M queries. The resulting attention matrix is interpreted as a soft heatmap $\mathbf{H} \in \mathbb{R}^{B\times M\times K}$.
2. **Self-attention over keypoints.** The updated queries are then passed through a second multi-head attention layer whose keys and values are themselves. This allows keypoints to exchange information and avoid redundant coverage.
3. **Keypoint refinement.** A two-layer feed-forward network with residual connection further refines each keypoint descriptor, analogous to the standard Transformer block.

The forward pass returns both the refined keypoint descriptors and the heatmaps. Keypoint coordinates and features are obtained via soft aggregation of sampled 3D points and descriptors using the attention weights.

Here MHA denotes multi-head attention with D-dimensional queries, keys, and values, and FFN is a two-layer MLP applied independently to each keypoint.

Visual–geometric fusion details. The visual–geometric fusion module is instantiated as a stack of feature aggregation blocks, each operating at a different neighborhood size $K \in \{8, 16, 32\}$. For each keypoint m with current feature $\mathbf{F}_m^{\text{obj}}$ and position $\mathbf{X}_m^{\text{obj}}$, a block proceeds as follows:

- **Neighborhood construction.** We perform KNN search in the predicted 3D point cloud to identify K neighbors for each keypoint. This is implemented with batched top-K over squared Euclidean distances.
- **Geometric encoding.** For each neighbor point $\mathbf{X}_k$, we encode (i) relative offset $\mathbf{X}_k - \mathbf{X}_m^{\text{obj}}$, (ii) absolute coordinate $\mathbf{X}_k^{\text{obj}}$, and (iii) long-range offsets to other keypoints.
- **Feature fusion.** The geometric encodings are concatenated with the per-point visual–geometric descriptor and projected by a extttfuse_mlp (two 1D convolutions with batch normalization and ReLU). A cosine-similarity attention with temperature τ is applied over the K neighbors to produce a weighted sum and a residual connection to update $\mathbf{F}_m^{\text{obj}}$.

Keypoint regularization. We detail the regularization terms introduced in the main paper. Let $\{\mathbf{X}_{i,m}^{\text{obj}}\}_{m=1}^{M}$ be the predicted keypoint set and $\{\mathbf{X}_{i,k}^{\star}\}$ be ground-truth surface samples obtained from depth. We approximate the one-sided Chamfer distance by computing, for each keypoint, the nearest neighbor in $\{\mathbf{X}_{i,k}^{\star}\}$ via batched KNN and averaging the squared distance. The diversity loss is implemented by explicitly forming all pairwise distances between keypoints in a frame and applying a soft hinge at radius $\tau_2 = 0.01$.

1.2. Canonical Correspondences and Absolute Pose

Latent-conditioned NOCS via bin classification. The NOCS head in OPT-Pose is instantiated as a FiLM-conditioned, bin-based regressor over canonical coordinates. Let K be the number of keypoints per frame and C the feature dimension. Given keypoint descriptors $\tilde{\mathbf{F}}^{\text{obj}} \in \mathbb{R}^{BS\times K\times C}$ (flattened over batch and sequence), the NOCS head performs the following:

- A three-layer 1D convolutional MLP with batch normalization and ReLU maps features back to dimension C, acting as a shared pre-processor for all keypoints.
- A self-attention layer further refines $\tilde{\mathbf{F}}^{\text{obj}}$ across keypoints within the same frame.
- For the FiLM conditioning, the object latent $\mathbf{z}_{\text{obj}} \in \mathbb{R}^d$ is projected by a linear layer into $(\gamma, \beta) \in \mathbb{R}^{2C}$, which modulates keypoint features via channel-wise affine transformation $x' = x \odot (1 + \gamma) + \beta$.
- A 1D convolution predicts logits of shape $[BS, 3B, K]$, where B is the number of discretization bins per axis. For each axis, we maintain a fixed table of B bin centers uniformly covering $[-0.5, 0.5]$. After a softmax over the bin dimension, the expected NOCS coordinate along each axis is obtained as a weighted sum over these bin centers.

SA(3) Pose Head initialization. The pose head that regresses anisotropic similarity transforms in $\mathrm{SA}(3)$. It en-

Capability	FoundPose [41]	MatchU [17]	VI-Net [32]	Oryon [7]	GCE-Pose [29]	AG-Pose [33]	Any6D [24]	OPT-Pose (Ours)
RGB Input	✓							✓
RGB-D Input		✓	✓	✓	✓	✓	✓	✓
Single view	✓	✓	✓		✓	✓		✓
Pair view				✓			✓	✓
Sequence								✓
Relative pose	✓			✓			✓	✓
Absolute pose		✓	✓		✓	✓		✓
Calibration-free								✓
CAD-free			✓	✓	✓	✓	✓	✓
Direct inference		✓	✓	✓	✓	✓		✓

Table 6. Comparison of representative methods in unseen object pose estimation. OPT-Pose is the only model-free framework supporting RGB and RGB-D input, single/pair/sequence views, both absolute SA(3) and relative SE(3) estimation, CAD-free and calibration-free operation, and direct feed-forward inference within a single model.

codes the two point sets (canonical NOCS keypoints and camera-frame keypoints) using separate 1D convolutional branches, concatenates them with keypoint features, and aggregates them with a global pooling operation. The final pose vector is obtained by an MLP and split into rotation, translation, and per-axis size components.

2. More Implementation Details

In this section, we describe training schedules, optimization details, data preprocessing, and evaluation protocols that complement Sec. 4 in the main paper. Unless otherwise stated, all hyperparameters are shared across datasets.

2.1. Training Setup

Training Hyperparameters and Losses. Our overall objective is a weighted multi-task loss consisting of camera, depth, point-map, NOCS, keypoint, pose, contrastive, metric-scale, and reconstruction terms. The camera loss has weight 5.0 and uses an ℓ_1 formulation. Depth and point-map losses both use weight 1.0 and include gradient supervision. The NOCS branch uses weight 2.0 with Smooth-ℓ_1 regression (threshold 0.1), together with keypoint-level regularization of weight 0.4, where the diversity and Chamfer terms are weighted by 5.0 and 2.0, respectively. For pose supervision, we use weight 1.0 with frobenius rotation loss, and set the relative weights of rotation, translation, and size to 1.0, 0.3, and 0.3. The supervised InfoNCE loss uses weight 1.0, temperature 0.1, and instance-consistency weight 0.05. For metric recovery, the absolute scale loss is weighted by 1.0, with equal weights on translation and size, while the RGB-based relative-scale loss, when enabled, uses weight 0.3 with Smooth-ℓ_1 parameter $\beta = 0.1$. Finally, the reconstruction loss uses weight 10.0, with an additional delta regularization weight of 1.0.

Hardware and Parallelism. We train OPT-Pose using Distributed Data Parallel (DDP) on 4 NVIDIA H100 GPUs. All experiments use mixed precision with bfloat16.

Optimizer and Learning Rate Schedule. We adopt AdamW as the optimizer. Following VGGT [55], we use separate parameter groups for the transformer backbone and task heads. The multiview aggregator and frozen backbone weights use a much smaller base learning rate (e.g., 5×10^{-7}) with weight decay 0.05, while the heads and projection layers (NOCS, pose, fusion modules, scale heads, and latent projectors) use a larger base learning rate (5×10^{-4}) with weight decay 10^{-2}. We apply 5% linear warm-up followed by cosine decay over the training schedule.

Gradient Clipping and Regularization. We employ per-module gradient clipping as configured in our training scripts. In practice, we set tighter clipping thresholds for the camera, depth, and point-map heads to stabilize geometry predictions, and looser thresholds for latent and fusion modules. Dropout is not used in the backbone but may be applied lightweightly in the heads.

2.2. Data Preprocessing and Augmentations

Image and Depth Preprocessing. All input RGB images are resized to 518×518 while preserving aspect ratio via padding where necessary. Depth maps are rescaled to metric units and masked outside the object region. We normalize depth and point maps to match the coordinate conventions used by VGGT [55], enabling direct reuse of pre-trained weights.

Object-centric cropping via camera warping. For the input cropped image, we construct a *virtual camera* that tightly focuses on the object while keeping its geometry consistent with the original camera model, which is crucial

because the geometry backbone expects the intrinsics with the principal point in the image center.

Concretely, given an original camera model $\mathcal{C}_{\text{orig}}$ with intrinsics K and pose $T_{\text{world}\leftarrow\text{eye}}$, and an object mask M, we first compute an amodal bounding box and then derive a crop box with optional padding. We constructs a new camera $\mathcal{C}_{\text{crop}}$ whose field of view covers the crop region and whose intrinsics are consistent with the model input resolution.

We form a sampling grid for extttgrid_sample by tracing rays from $\mathcal{C}_{\text{crop}}$ back to the original camera:

1. For each destination pixel in the crop, unproject a ray using the inverse of K_{crop}.
2. Transform these rays into world coordinates using $T^{\text{crop}}_{\text{world}\leftarrow\text{eye}}$, then into the original camera frame using $(T^{\text{orig}}_{\text{world}\leftarrow\text{eye}})^{-1}$.
3. Project the resulting 3D directions with K_{orig} to obtain the source pixel locations.

The same grid is reused to warp RGB, NOCS, masks, and depth/sensor-depth into the crop view, and we extract the updated intrinsics K_{crop} from $\mathcal{C}_{\text{crop}}$. This ensures that VGGT always receives object-centric images together with intrinsics whose principal point lies near the image center, while fully preserving metric consistency between image, depth, and NOCS.

3. More Results and Visualization

We showcase more visualization of relative pose estimation in Fig. 5 and absolute category-level pose estimation on NOCS-Real Fig. 6 and Housecat6D Fig. 7.

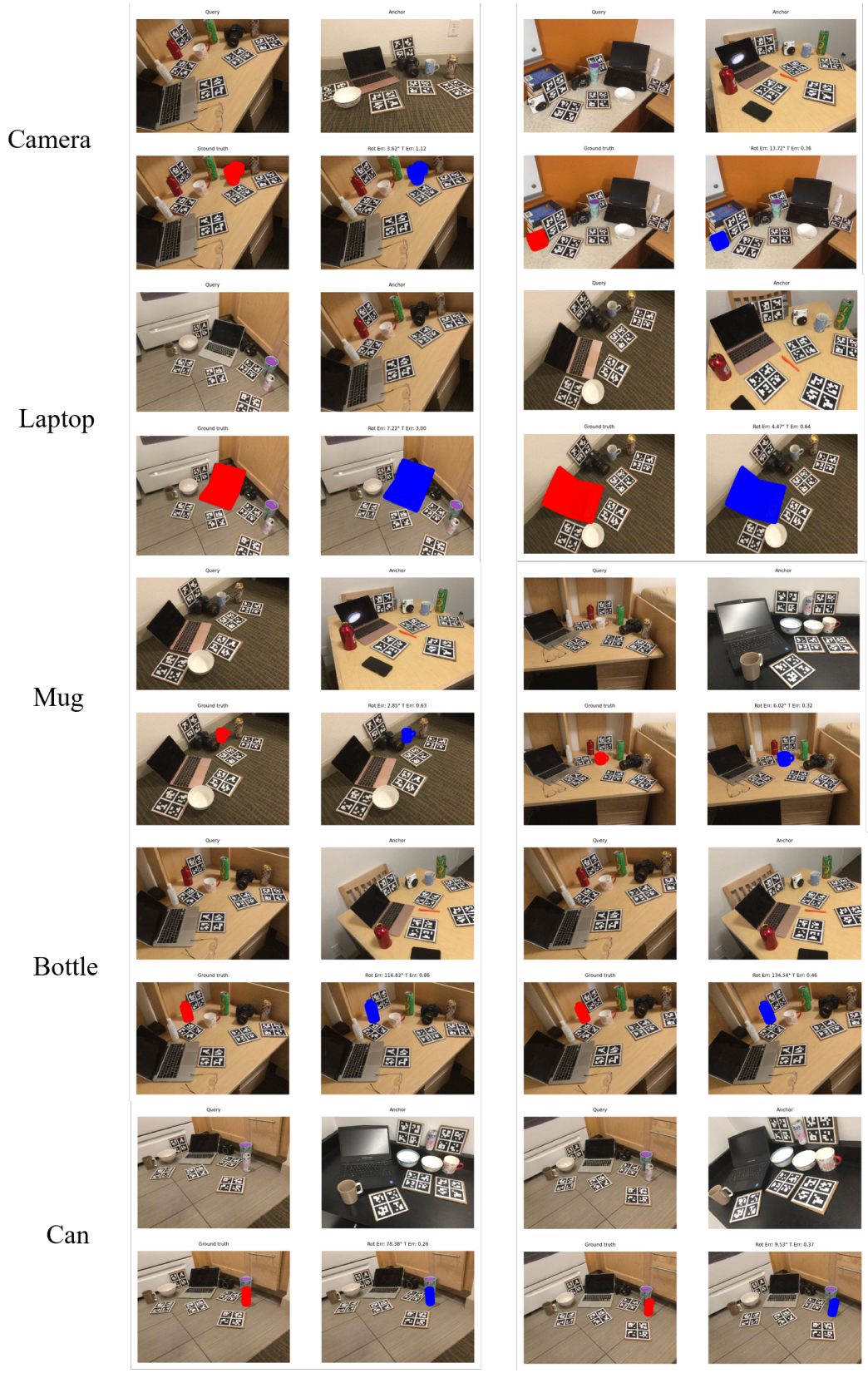

Figure 5. Qualitative Result: Unseen Relative Object Pose Estimation on NOCS-Real dataset.

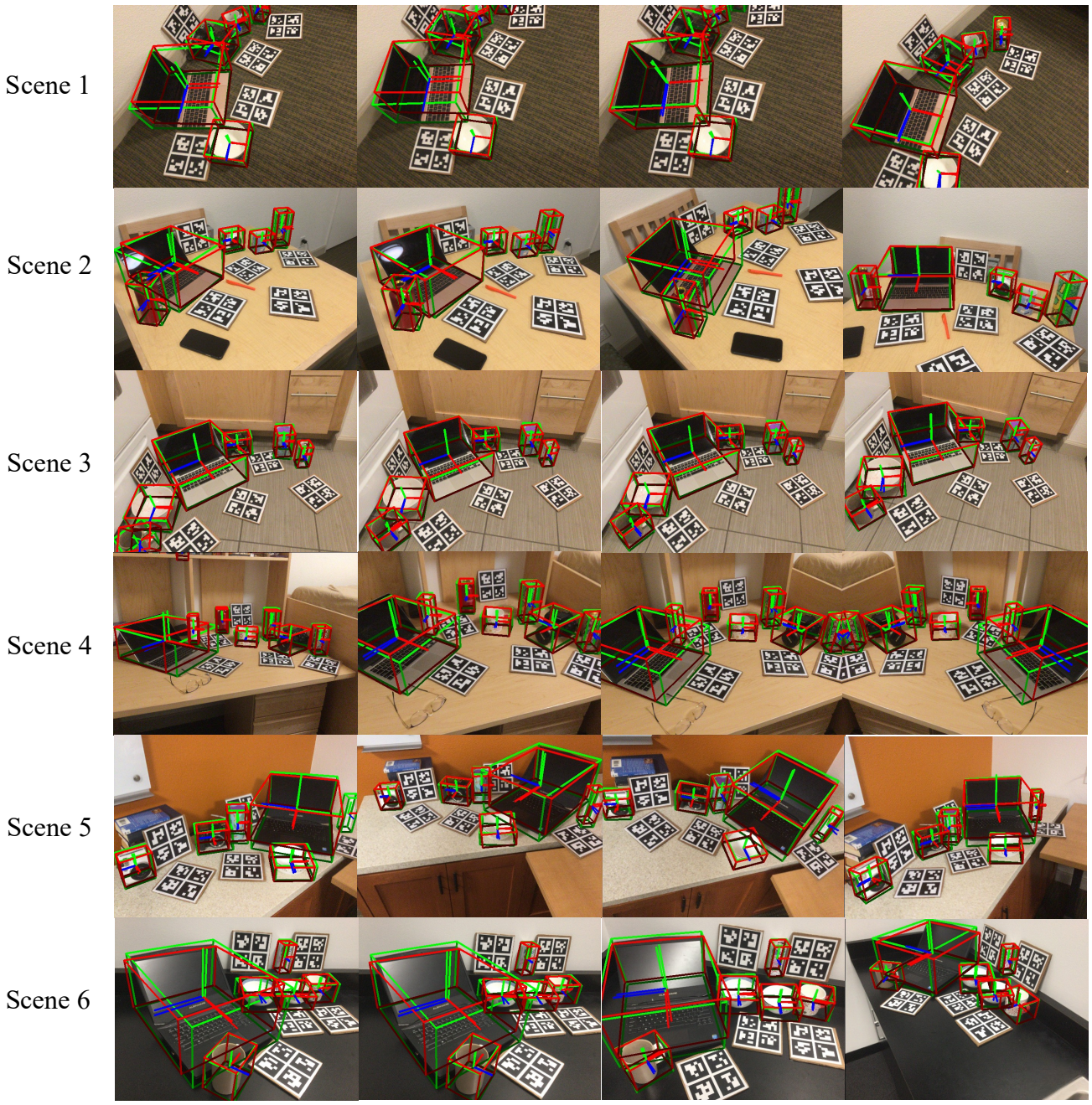

Figure 6. Qualitative Result: Category-level Absolute Object Pose Estimation on NOCS-Real dataset.

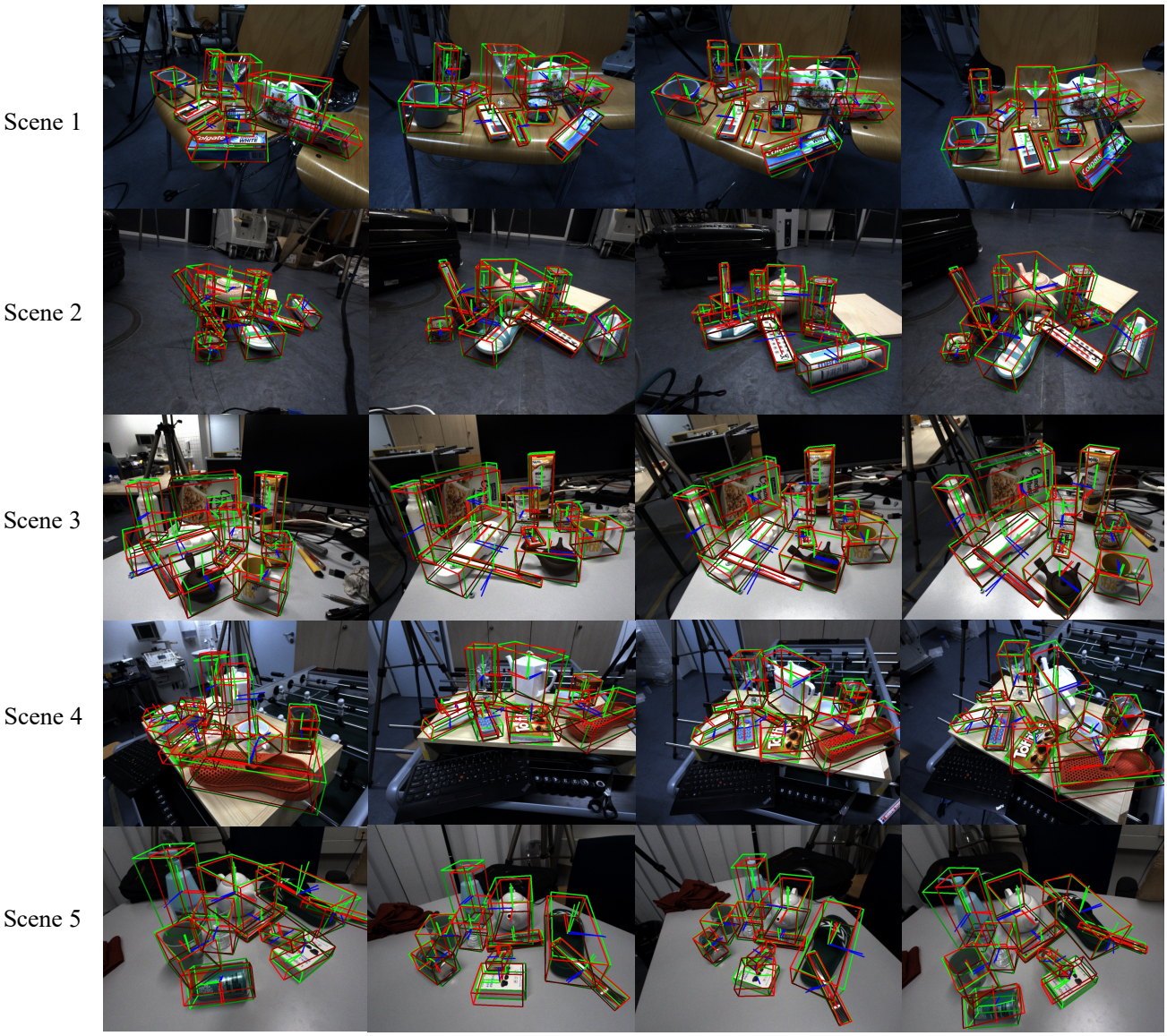

Figure 7. Qualitative Result: Category-level Absolute Object Pose Estimation on Housecat6D dataset.